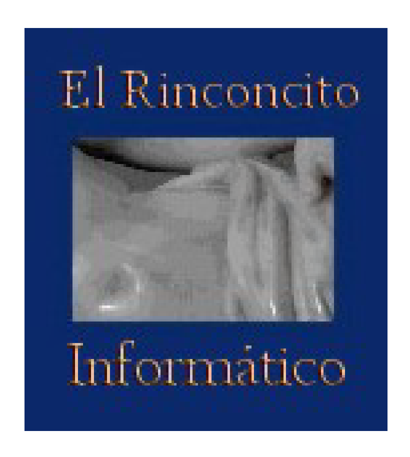
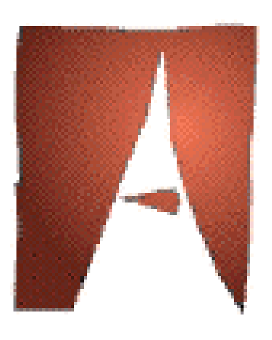


Seguridad en Redes IP

Gabriel Verdejo Alvarez



Universitat Autònoma de Barcelona

Departament d'Informàtica
Unitat de Combinatòria i Comunicació Digital (CCD)

SEGURIDAD EN REDES IP

Memoria del trabajo de investigación correspondiente a los estudios de doctorado en ***Informática,*** realizado en el Departamento de Informática y presentado por **Gabriel Verdejo Alvarez**.

Bellaterra, Septiembre del 2003.

El firmante, Sr. Joan Borrell Viader, profesor del departamento de informática de la Universidad Autónoma de Barcelona certifica:

Que la presente memoria ha sido realizada bajo su dirección por Gabriel Verdejo Alvarez.

Bellaterra, Septiembre del 2003.

Firmado: Joan Borrell Viader

"*Se acabó la coartada de los últimos cinco años. Ya no seremos universitarios. La coartada anterior, la de que éramos niños, queda también descartada por razones obvias. Tienes miedo porque ves que tu vida va a cambiar hacia algo que no conoces. Dentro de nada nos echaran de esta guardería para parados que es la universidad.*

Yo por mi parte pienso ponérselo difícil. Me voy a doctorar y voy a intentar como sea entrar en la pandilla."

Carlos Bardem (Muertes ejemplares)

"The only system which is truly secure is one which is switched off and unplugged, locked in a titanium lined safe, buried in a concrete bunker, and is surrounded by nerve gas and very highly paid armed guards. Even then, I wouldn't stake my life on it."

Gene Spafford

A todos aquellos que me han ayudado durante toda mi vida. A los demás, no.
A Meritxell, por todo. Ya tienes tu nombre en dos libros.

ÍNDICE

Introducción

En este trabajo de investigación denominado "**Seguridad en redes IP**" se pretende dar una visión profunda de los riesgos existentes en las redes conectadas a Internet así como de los principales sistemas de seguridad existentes.

Internet, denominada también la red o red de redes, se ha convertido en la última década en un fenómeno que ha revolucionado la sociedad. Desde la aparición de la televisión no se ha observado ningún otro fenómeno social de tal envergadura o que evolucione tan rápido.

De los muchos factores que han convergido en este nuevo fenómeno para catapultarlo de forma masiva a la sociedad actual, podemos destacar tres principalmente:

- La expansión de los ordenadores en todos los ámbitos de la sociedad (empresas, universidades, gobiernos…) ha contribuido a informatizar casi cualquier aspecto de nuestra vida.

- La rápida evolución de la tecnología de las comunicaciones (más rápido, más barato, mejor) ha acelerado aún más el despegue de Internet.

- El carácter universal de Internet que permite la conectividad global y permanente de todo el planeta de forma económica y prácticamente instantánea, lo convierten en una herramienta imprescindible para prácticamente cualquier tipo de comunicación.

En consecuencia, cada día cientos de millones de personas en todo el mundo utilizan Internet como parte de su trabajo y ocio. De igual forma que en cualquier otro servicio utilizado por gran cantidad de personas (como el metro o las carreteras), la seguridad es un factor básico que siempre debe ser tenido en cuenta.

Seguridad (*l. securitate*) : **1** *f.* Calidad de seguro [WWW1].
Seguro, -ra (l. *securu*) : **1** *adj.* Exento de todo peligro o riesgo [WWW1].

Desde un punto de vista sociológico, en cualquier grupo social un pequeño porcentaje de su población es malévolo [CZ95]. Internet ha alcanzado en el año 2002 más de 160.000.000 de ordenadores conectados [WWW5] sumando un total estimando de 580.000.000 de usuarios en todo el mundo [WWW6][WWW7]. Si tan sólo el 1 por cien de la población pertenece a este sector tenemos casi 6 millones (5.800.000) de posibles atacantes. Incluso suponiendo sólo un uno por mil tenemos la cantidad de casi seiscientos mil (580.000) peligros potenciales.

El objetivo de este estudio se centra precisamente en el análisis de los peligros y amenazas más comunes que existen en Internet así como en los nuevos mecanismos de seguridad que pretenden darles solución.

1.1 Resumen de los capítulos

La estructura que presenta este trabajo se divide en cinco capítulos que se agruparán en dos bloques. La primera parte hace referencia al trabajo de investigación teórica y se compone de los primeros cuatro capítulos, mientras que la segunda parte se compone del capítulo cinco dónde se presenta la parte experimental realizada.

1. **Los protocolos IP**: En el primer capítulo se realiza una explicación en profundidad de la familia de protocolos TCP/IP versión 4. Se describen las características y funcionalidades más importantes que presentan los protocolos IP, ICMP, UDP y TCP así como su interrelación y papel que juegan en la comunicación de sistemas conectados a Internet.

   También se explica el proceso de traslado de los datagramas por Internet (rutado de paquetes) hasta llegar a la red local de destino (LAN), dónde es entregado al ordenador especificado por la dirección IP gracias a los protocolos ARP/RARP.

2. **Denegación de servicio (DOS / DDOS)**: En este segundo capítulo realizaremos un repaso histórico de la evolución de los ataques a redes de ordenadores en Internet, centrándonos en el estudio y clasificación de los ataques de denegación de servicio o DOS.

   Los ataques DOS son aquellos destinados a conseguir de forma total o parcial el cese de un servicio existente. Estos ataques se basan en el uso de diferentes técnicas que intentan colapsar el servicio en sí mismo o el ordenador que lo soporta mediante una inundación de peticiones fraudulentas.

   Los ataques DOS distribuidos (DDOS) se caracterizan por la sincronización de varios ordenadores distintos que focalizan sus ataques de forma coordinada hacia un mismo destino.

   En la parte final de este capítulo realizaremos un análisis exhaustivo de un ataque DDOS real registrado en Internet el 11 de enero de 2002.

3. **Sistemas de detección de intrusos** (IDS): En este tercer capítulo realizaremos una explicación de los sistemas de detección de intrusos o IDS. Comentaremos la taxonomía en la que se dividen los sistemas de detección de intrusos: sistemas de ordenador (HIDS) y sistemas de red (NIDS) para centrarnos en estos últimos.

   Los sistemas NIDS son una evolución de los primitivos firewalls que únicamente filtraban el tráfico de red existente entre Internet y la red LAN. Entre sus nuevas capacidades añaden la de analizar el tráfico existente en toda la red local en búsqueda de anomalías o comportamientos sospechosos.

   Analizaremos también sus protocolos de comunicación, sus posibles ubicaciones y arquitecturas así como las limitaciones que pueden presentar. También se introducirán los conceptos de falsos positivos y falsos negativos que hacen referencia a las detecciones erróneas que a veces producen estos sistemas o a la no detección de un comportamiento extraño por el IDS.

   Finalmente se describirá de forma minuciosa el "ataque Mitnick" perpetrado por uno de los *hackers* más famosos del mundo Kevin Mitnick y que le costó varios años de cárcel. En la actualidad este ataque clásico se considera el umbral mínimo de detección para un sistema IDS.

4. **Honeypots y Honeynets**: En el último capítulo de la parte de investigación se describirá el nuevo escenario que se está produciendo como consecuencia de la evolución de las comunicaciones en Internet.

   El abaratamiento de los costes de conexión y el aumento del ancho de banda disponible modifican los escenarios típicos de ataques. Consecuentemente la comunidad investigadora propone una nueva herramienta de seguridad: los Honeypot (potes de miel textualmente o ratoneras).

   Los Honeypots son sistemas pasivos cuyo funcionamiento se basa en estar diseñados para ser atacados e incluso comprometidos por cualquier atacante. El objetivo de tener un sistema destinado a ser atacado es doble:

Por un lado permitir el estudio de los comportamientos y técnicas reales que utilizan los *hackers* en un entorno "real". Este entorno puede ser configurado de forma que incluso pueda ser capaz de proporcionar información falsa a eventuales atacantes.

Por otro lado, la existencia de un sistema con estas características nos permite desviar la atención sobre nuestros sistemas reales y prepararlos para los ataques registrados en el Honeypot.

Las Honeynets son un tipo concreto de Honeypots. El objetivo es la existencia de diferentes Honeypots agrupadas en una red "aislada" de la red de producción con el objetivo de crear un entorno más verosímil para los posibles atacantes.

Comentaremos las dos generaciones (GEN I y GEN II) de Honeynets existentes así como sus características y arquitecturas principales. También introduciremos los conceptos de Honeynets virtuales y distribuidas que permiten la minimización de los recursos necesarios para la implementación de estas técnicas.

5. **Análisis de un sistema conectado a Internet**: En el quinto capítulo realizamos la parte experimental de este trabajo. Nuestro objetivo es el de monitorizar durante siete días un sistema conectado permanentemente a Internet.

   Analizaremos los distintos objetivos que plantea la tarea exigida y expondremos los distintos requerimientos del experimento. Se propondrá una arquitectura de red que cumpla nuestros requisitos y se escogerán las distintas herramientas (software y hardware) que nos permitirán evaluar nuestro experimento.

   La presentación de los datos se realizará mediante un informe diario pormenorizado con los datos del tráfico de red obtenidos (distintas peticiones y ataques recibidos) que se desglosará en tráfico registrado por tipo de servicio (SSH, WWW…) y por tipo de protocolo al que hace referencia.

Finalmente también se presentará un informe semanal que contendrá los aspectos más relevantes registrados durante los siete días analizados así como las conclusiones obtenidas del experimento.

En la parte final de este trabajo se presentarán las conclusiones obtenidas durante la realización de este informe así como las futuras líneas de continuación que podrían llevarse a cabo.

CAPITULO 1

Los protocolos TCP/IP

En este primer capítulo realizaremos un estudio profundo de los protocolos básicos que permiten el funcionamiento de Internet. La familia de protocolos TCP/IP es la piedra angular sobre la que se sustentan el resto de servicios (WWW, FTP, SSH...) que actualmente podemos encontrar en Internet, y por tanto su estudio es parte fundamental para el resto del trabajo.

Se introducirá el sistema de direccionamiento, el formato utilizado en sus cabeceras y el modo de funcionamiento de los protocolos más importantes de esta familia:

- El protocolo de control de flujo (**ICMP**).

- El protocolo no fiable de transmisión de datos sin conexión (**UDP**).

- El protocolo fiable de transmisión de datos con conexión (**TCP**).

Finalmente se explicará el proceso de rutado (*routing*) de los datagramas IP así como su transito por la red local hasta el ordenador destinatario.

Este estudio se centrará únicamente en la versión 4 del protocolo IP, ya que aunque actualmente se está trabajando en la versión 6 [Ver00] esta es aún experimental.

1.1 Redes IP

Definiremos las **redes IP** como aquellas redes que utilizan los protocolos TCP/IP para su funcionamiento. Internet es una red IP.

"Las familias de protocolos TCP/IP permiten la comunicación entre diferentes tipos de ordenadores con independencia del fabricante, red a la que se encuentren conectados y sistema operativo utilizado." [Ric98-1]

Las redes IP se caracterizan por haber sido construidas siguiendo un esquema de capas (*layers*). Cada capa es la responsable de cada una de las diferentes facetas de la comunicación. De esta forma, se puede definir la familia de protocolos TCP/IP como una combinación de cuatro capas (ver figura 1-1) según el modelo OSI [Ric98-1].

En este esquema, la capa superior accede únicamente a los servicios prestados por la capa situada justo en el nivel inferior a ella. De esta forma, independizamos una capa del resto de capas inferiores, lo que nos permite tener un esquema modular.

APLICACIONES (Applications)	WWW, SSH, FTP...
TRANSPORTE (Transport)	TCP, UDP, ICMP...
RED o Interconexión de redes (Network)	IP
ENLACE o Red real (Link)	IEEE 802.2, 802.3...

FIG. 1-1: Estructura de cuatro capas.

- La ***capa de enlace*** o **capa de red real** (*link layer*), también denominada la capa de datos (*data layer*) o capa de acceso a red (*network interface layer*), incluye los mecanismos que permiten al sistema operativo enviar y recibir información a través de la red a la que se encuentra físicamente conectado (Ethernet, RDSI...).

- La ***capa de red*** o **capa de interconexión de redes** (*network layer),* también denominada capa de Internet (*Internet layer*), es la encargada de mover los paquetes de información a través de las diferentes redes para llegar a su destino. En esta capa encontramos los protocolos de más bajo nivel, destacando el IP (*Internet Protocol*).

- La ***capa de transporte*** (*transport layer*), es la encargada de proporcionar un flujo de datos entre dos ordenadores. Este flujo de datos puede ser fiable (*Transmission Control Protocol, TCP*) o no fiable (*User Datagram Protocol, UDP*).

- La ***capa de aplicaciones*** (*applications layer*), es la encargada de manejar los detalles particulares relativos a las diferentes aplicaciones que utilizará el usuario (WWW, TELNET, FTP...).

Este sistema permite una independencia entre las diferentes capas y obliga a que la comunicación entre dos ordenadores se realice mediante una comunicación entre las capas del mismo nivel de los dos ordenadores (ver figura 1-2).

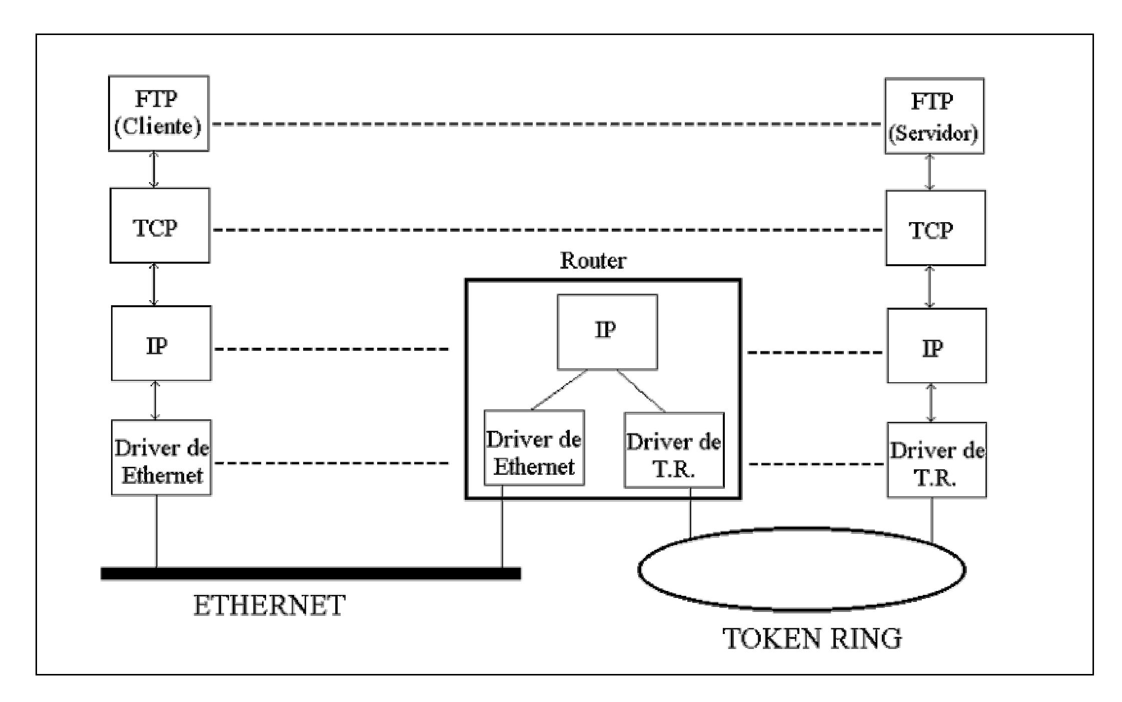


FIG. 1-2: Esquema de conexión de dos ordenadores en Internet.

La comunicación en Internet se produce mediante el intercambio de paquetes de información entre los distintos ordenadores. Estos paquetes de información (también denominados **datagramas**) viajan por los diferentes ordenadores que están conectados a Internet hasta que alcanzan su objetivo o son descartados por algún motivo.

De esta forma, en la comunicación de dos ordenadores por Internet podemos diferenciar dos tipos de funciones que pueden desempeñar los ordenadores por los cuales se transmiten los paquetes de información:

1. Ordenador ***emisor/receptor*** (*end-system o end-host*): Aquí se englobaría el ordenador origen o destinatario de la comunicación.

2. Ordenador ***intermedio*** *(intermediate-system, router o gateway*): Serían todos los ordenadores por los que van pasando los datagramas o paquetes de información hasta el ordenador destino de la comunicación o hasta el origen (en el caso de una respuesta).

El protocolo IP dispone de un sistema de numeración que permite diferenciar todos y cada uno de los ordenadores conectados. En la versión 4 de los protocolos TCP/IP[ℜ], estas direcciones han de cumplir dos requisitos básicos (ver figura 1-3):

1. Deben ser únicas. No puede haber dos ordenadores con la misma dirección.

2. Las direcciones son números de 32 bits (4 bytes). Estas direcciones se representan mediante cuatro números decimales separados por un punto.

CLASE	**RANGO**		
A	0.0.0.0	Hasta	127.255.255.255
B	128.0.0.0	Hasta	191.255.255.255
C	192.0.0.0	Hasta	223.255.255.255
D	224.0.0.0	Hasta	239.255.255.255
E	240.0.0.0	Hasta	247.255.255.255

FIG. 1-3: Clases de direcciones IP en Internet

Este tipo de direccionamiento, nos permite una gran flexibilidad a la hora de definir redes que posteriormente conectaremos a Internet. Así, una clase A sería ideal para redes muy grandes, ya que permite 128 redes (2^7) de 16.777.216 (2^24) ordenadores cada una. Mientras que una clase B permite 16.384 (2^14) redes con 65.535 ordenadores, y una clase C permite 2.097.152 (2^21) redes de 256 ordenadores.

Las clases D (*multicast*) y E (*reservada*) se utilizan para diferentes posibilidades como la de tener ordenadores en redes diferentes y que se vieran como si estuvieran en la misma (Ej. 2 ordenadores en la UAB, 1 en la UPC y 10 en la UB, y que todos recibieran la misma información, como en una multi-conferencia).

No obstante estas particularidades van más allá de los propósitos de este trabajo, y se recomienda la lectura de [Ric98-1] para más información.

[ℜ] Aunque ya se llevan años trabajando en la versión 6 del protocolo IP [Ver00] la versión 4 es la que se está utilizando actualmente.

Una vez definido el direccionamiento de redes y ordenadores en Internet, mencionaremos la existencia de los servicios de DNS (*Domain Name Server*).

Debido a que es más fácil recordar un nombre (Centro de cálculo de la Universidad Autónoma de Barcelona, cc.uab.es) que una dirección numérica (158.109.0.4), se crearon los servidores de nombres (DNS), que son las máquinas encargadas de transformar un nombre en su dirección correspondiente.

1.2 El protocolo IP versión 4

El protocolo IP (*Internet Protocol*) es la pieza fundamental en la que se sustenta el sistema TCP/IP y por tanto todo el funcionamiento de Internet. Su especificación está recogida en **[RFC791]**.

La unidad de datos del protocolo IP es el *datagrama* (ver figura 1-4), cuyo tamaño máximo es de 65535 bytes (64K).

El protocolo IP facilita un sistema **sin conexión** (*connectionless*) y **no fiable** (*unreliable*) de entrega de datagramas entre dos ordenadores cualesquiera conectados a Internet.

IP da un servicio de entrega basado en el mejor intento (*best effort*). Esto implica que cuando hay algún funcionamiento anómalo de Internet, como podría ser un *router* colapsado, se contempla un sistema muy simple de tratamiento de errores. Este mecanismo de control de errores viene regulado por el protocolo ICMP (*Internet Control Message Protocol*).

En nuestro caso, el router colapsado descartaría el datagrama y enviaría un mensaje de error ICMP al ordenador de origen sin encargarse de la retransmisión del datagrama, lo que **no implica fiabilidad**.

Además, no mantiene ningún tipo de información referente al estado de las conexiones. Cada datagrama es encaminado de forma independiente, lo que le convierte en un **protocolo sin conexión**.

Debido a estas particulares características, puede pasar que se pierdan datagramas y/o que estos no lleguen en orden. De esta manera, cualquier fiabilidad que se necesite, deberá ser realizada por las capas superiores (TCP...).

La estructura de un datagrama IP está dividida en bloques de 32 bits (4 bytes). El datagrama IP se transmite enviando primero el bit 0, luego el bit 1, 2, 3... y así sucesivamente hasta finalizar el datagrama.

Este orden se denomina ***network byte order***. Es muy importante conocer este orden de transmisión de la información, puesto que los diferentes ordenadores tienen diferentes sistemas de almacenamiento de bits en memoria.

El formato *little endian*, consiste en almacenar los bits en orden inverso al *network byte order* (usando por ejemplo en los procesadores Intel), mientras que la otra posibilidad se denomina *big endian* (usado por ejemplo en los procesadores Motorola).

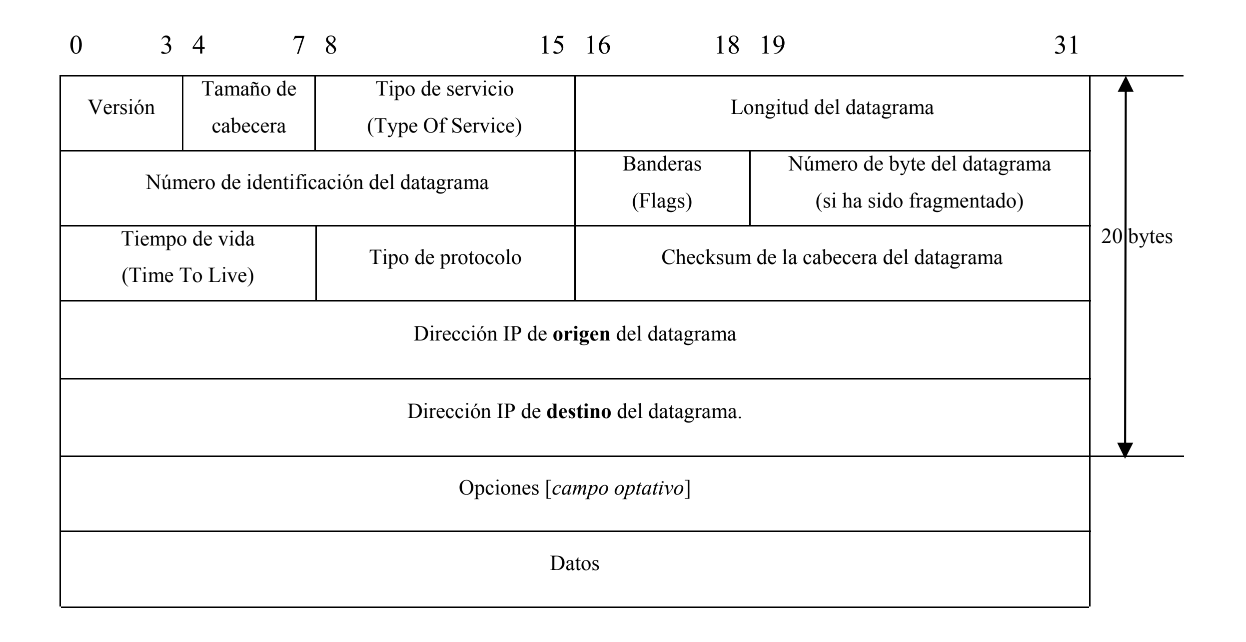


FIG. 1-4: Estructura de un datagrama IP v4.

La **versión** (4 bits), sirve para identificar a que versión específica (RFC) hace referencia el formato del datagrama. Esta información sólo es utilizada por los routers y capa IP de origen y final del datagrama. Esto permite la coexistencia de diferentes versiones del protocolo IP de una forma transparente al usuario. La versión actual es la 4 (conocida también cómo IPv4).

El **tamaño de la cabecera** (*Header Length*), son 4 bits (2^4 = 16 posiciones, 0...15) que indican el número de palabras de 32 bits que ocupa la cabecera. Estos 4 bits de tamaño máximo nos limitan a un tamaño de cabecera máximo de 60 bytes (15 * 32 bits = 60 bytes). No obstante, el valor usual de este campo es 5 (5 * 32 bits = 20 bytes).

El **campo del tipo de servicio** (*Type Of Service*), se compone de 8 bits. Los primeros 3 bits tienen una función obsoleta y no se contemplan actualmente. Los 4 bits siguientes definen el tipo de servicio (ver figura 2-12) y el último bit no se utiliza actualmente y debe tener valor 0. Solo 1 de los 4 bits del tipo de servicio puede estar activo a la vez. El tipo de servicio determina la política a seguir en el envío del datagrama por Internet. Las opciones posibles son:

1. minimizar el retraso (*minimize delay*)
2. maximizar el rendimiento (*maximize throughput*)
3. maximizar la fiabilidad del transporte (*maximize reliability*)
4. minimizar el coste económico del transporte (*minimize monetary cost*).

Tipo de aplicación	**Minimizar retraso**	**Maximizar rendimiento**	**Maximizar fiabilidad**	**Minimizar coste**	**Valor en hexadecimal**
TELNET	1	0	0	0	0x10
FTP	0	1	0	0	0x08
SMTP	0	1	0	0	0x08
DNS (UDP)	1	0	0	0	0x10
DNS (TCP)	0	0	0	0	0x00
ICMP	0	0	0	0	0x00
BOOTP	0	0	0	0	0x00

FIG. 1-5: Valores típicos del tipo se servicio según la aplicación.

La **longitud del datagrama** (*Total Length*), es un número de 16 bits (2^16 = 65536, 0...65535) que indica la longitud total del datagrama. Este valor es muy importante, ya que nos permite saber que tamaño de memoria debemos reservar para la recepción del datagrama. Además, nos indica el número de bytes a leer, lo que nos permite un simple control de error. De esta forma, si el valor es incorrecto, el número de bytes leídos será como máximo de 65535, acotando el error. Además nos limita el número de bytes a enviar en un datagrama (*Maximum Transfer Unit, MTU*) a 65535 – 20 (tamaño típico de la cabecera) = 65515 bytes.

Si el tamaño del datagrama, es mayor que el tamaño máximo del paquete de red (Ej. Datagrama de 32000 bytes enviado sobre una Ethernet, que tiene un tamaño máximo de paquete de 1500 bytes), éste se fragmenta en N trozos.

El **número de identificación del datagrama** (*Identificartion Field*), es un número de 16 bits que en caso de fragmentación de un datagrama nos indica su posición en el datagrama original. Esto nos permite recomponer el datagrama original en la máquina de destino. Este valor nos indica que un datagrama puede ser fragmentado en un máximo de 65535 fragmentos.

Las **banderas** (*Flags*) son 3 bits. El primero permiten señalar si el datagrama recibido es un fragmento de un datagrama mayor, bit M (*More*) activado. El segundo especifica si el datagrama no debe fragmentarse, bit DF (*Don't fragment*) activado y el tercero no se utiliza actualmente, asignándole el valor 0 [San99].

El **número de byte en el datagrama** (*Fragmentation Offset*), nos indica la posición en bytes que ocupan los datos en el datagrama original. Sólo tiene sentido si el datagrama forma parte de uno mayor que ha sido fragmentado. Este campo tiene un máximo de 13 bits (2^13 = 8192, como nos indica el desplazamiento en bytes 8192 * 8 bits = 65536). De esta forma, siempre podemos reconstruir el datagrama original con los fragmentos.

El **tiempo de vida** (*Time To Live*), es un campo de 8 bits que indica el tiempo máximo que el datagrama será válido y podrá ser transmitido por la red. Esto permite un mecanismo de control para evitar datagramas que circulen eternamente por la red (por ejemplo en el caso de bucles).

Este campo se inicializa en el ordenador de origen a un valor (máximo 2^8 = 256) y se va decrementando en una unidad cada vez que atraviesa un router. De esta forma, si se produce un bucle y/o no alcanza su destino en un máximo de 255 "saltos", es descartado. En este caso se envía un datagrama ICMP de error al ordenador de origen para avisar de su pérdida.

El **tipo de protocolo** (*Protocol*), es un valor que indica a que protocolo pertenece el datagrama (TCP, UDP, ICMP...). Es necesario debido a que todos los servicios de Internet utilizan IP como transporte, lo cual hace necesario un mecanismo de discriminación entre los diferentes protocolos.

El **checksum de la cabecera del datagrama** (*Header Checksum*), es una suma de comprobación que afecta sólo a la cabecera del datagrama IP. El resto de protocolos TCP, UDP, IGMP... tienen su propia cabecera y *checksum*. Su función es simplemente la de un mecanismo de control de errores. De esta forma, si se encuentra un error en el *checksum* de un datagrama IP, este es simplemente descartado y no se genera ningún mensaje de error. Esto implica que es deber de las capas superiores el control del flujo de los datagramas para asegurarse que estos lleguen correctamente al destino, ya sea utilizando un protocolo fiable (TCP) o implementando internamente algún tipo de control.

Tanto la **dirección IP de origen como la de destino** (*IP address*), están formadas por dos números de 32 bits. Estas direcciones se corresponden a una distribución según la figura 1-3.

1.3 El protocolo ICMP

El protocolo ICMP (*Internet Control Message Protocol*) es un protocolo simple que va encapsulado en datagramas IP (ver figura 1-6) y que tiene por función el control del flujo de la comunicación así como la comunicación de errores [RFC792].

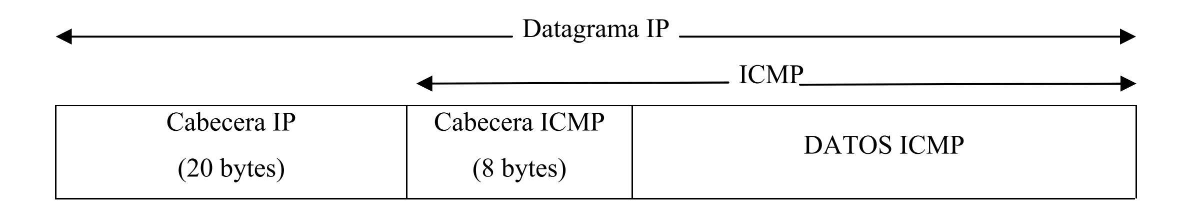


FIG. 1-6: Estructura de un mensaje ICMP.

La cabecera del protocolo ICMP tiene un tamaño de 8 bytes que contiene varios campos (ver figura 1-7) que permiten la identificación del mensaje.

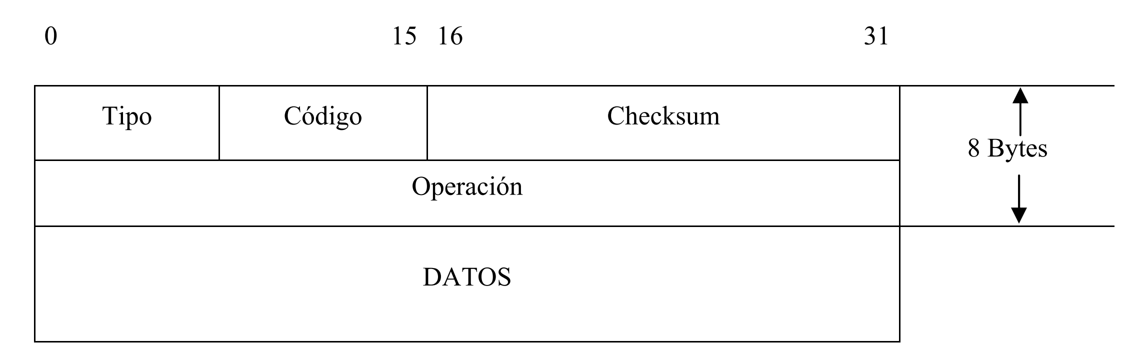


FIG. 1-7: Cabecera de un datagrama ICMP.

El **Tipo** (*Type*) indica el carácter del mensaje enviado, ya que el protocolo permite especificar una gran variedad de errores o mensajes de control de flujo de las comunicaciones (ver figura 1-8).

El campo de **Código** (*Code*) indica el código de error dentro del tipo de error indicado en el campo "tipo". Es decir, agrupa los mensajes en tipos y dentro de cada tipo especifica el código concreto al que se refiere (ver figura 1-8).

El **Checksum** simplemente permite verificar la integridad del mensaje enviado, lo que permite detectar posibles errores en el envío, transporte o recepción del mensaje de control ICMP.

El campo de **Operación** (*Operation*) depende directamente del contenido de los campos de tipo y código, permitiendo la inclusión de una información extra referida al código de error.

TIPO (*Type*)	**CÓDIGO** (*Code*)
0	Echo Reply Codes: 0 No Code
1	Unassigned
2	Unassigned
3	Destination Unreachable Codes: 0 Net Unreachable 1 Host Unreachable 2 Protocol Unreachable 3 Port Unreachable 4 Fragmentation Needed and Don't Fragment was Set 5 Source Route Failed 6 Destination Network Unknown 7 Destination Host Unknown 8 Source Host Isolated 9 Communication with network is Administratively Prohibited 10 Communication with Host Administratively Prohibited 11 Destination Network Unreachable for TOS 12 Destination Host Unreachable for TOS
4	Source Quench Codes: 0 No Code
5	Redirect Codes: 0 Redirect for the Network 1 Redirect Datagram for the Host 2 Redirect for the TOS and Network 3 Redirect for TOS and Oct
6	Alternate Host Address Codes: 0 Alternate Address for Oct
7	Unassigned
8	Echo Codes: 0 No Code
9	Router Advertisement Codes: 0 No Code
10	Router Selection Codes: 0 No Code
11	Time Exceded Codes: 0 Time to Live exceeded in Transit 1 Fragment Reassembly Time Exceded
12	Parameter Problem Codes: 0 Pointer indicates the error 1 Missing a Required Option 2 Bad Length
13	Timestamp Codes: 0 No Code
14	Timestamp Reply Codes: 0 No Code
15	Information Request Codes 0 No Code
16	Information Reply Codes 0 No Code
17	Address Mask Request Codes 0 No Code
18	Address Mask Reply Codes: 0 No Code
19	Reserved (for Security)
20-29	Reserved (for Robustness Experiment)
30	Traceroute
31	Datagram Conversion Error
32	Mobile Host Redirect
33	IPv6 Where-Are-You
34	IPv6 I-Am-Here
35	Mobile Registration Request
36	Mobile Registration Reply

FIG. 1-8: Tabla de tipos y códigos del protocolo ICMP.

1.4 El protocolo UDP

El protocolo UDP (*User Datagram Protocol*) se podría definir como un **protocolo simple** y **orientado a datagrama** [Ric98-1]. Su definición se recoge en **[RFC7680]** publicado por Postel en 1980.

De esta forma, cada envío de datos se corresponde con un único envío de un datagrama independiente del resto de datagramas y de la misma comunicación (ver figura 1-9).

De esta forma, siguiendo los preceptos de encaminamiento de datagramas por Internet, la entrega al destino no está asegurada por el propio protocolo. Es mas, ni tan siquiera se asegura que los datagramas lleguen en el orden en el que fueron enviados.

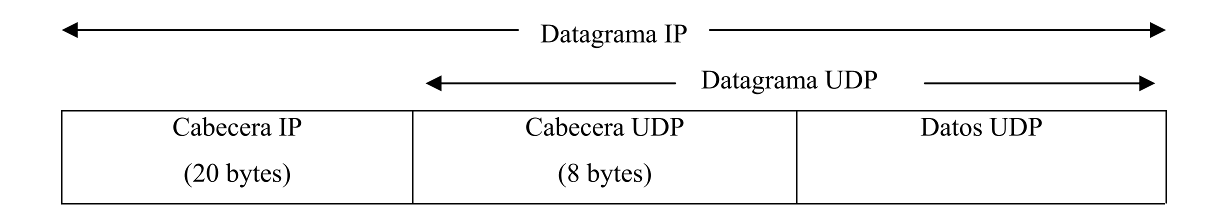


FIG. 1-9: Estructura de un datagrama UDP.

Debido a que UDP utiliza IP para su transporte por Internet, en caso de ser necesario (por diferentes tamaños de MTU, por ejemplo), este se fragmentará y ninguno de estos fragmentos (al igual que el datagrama original) proporcionara ningún tipo de seguridad o fiabilidad en la entrega.

Debido a la sencillez de este protocolo, el diseño de su cabecera (figura 1-10), es mucho más simple que el de IP.

El **número de puerto** (*Port*) se utiliza en la comunicación entre dos ordenadores para diferenciar las diferentes conexiones existentes. Si tenemos varias comunicaciones desde nuestro ordenador (por ejemplo un TELNET al *cc.uab.es* mirando el correo y un

FTP a *blues.uab.es* bajando un fichero), al recibir un datagrama IP debemos saber a cual de las conexiones pertenece.

Asignando un número de puerto a la comunicación podemos saber a que conexión pertenece. Al ser un número de 16 bits, podemos deducir que el número máximo de conexiones que un ordenador puede tener simultáneamente en uso es de 65535 (2^16).

En [WWW58] se puede obtener una lista completa de los servicios asociados a los puertos TCP y UDP.

La **longitud del datagrama** (*UDP Length*) hace referencia al tamaño del datagrama en bytes, y engloba la cabecera (8 bytes) más los datos que transporta. El mínimo valor de longitud es 8 bytes (por lo tanto, el protocolo permite enviar un datagrama UDP con 0 bytes).

Este campo es redundante, ya que utiliza IP para su transporte, y éste ya incorpora un campo para la longitud de los datos (ver figura 2-11) que sería la longitud del datagrama IP menos el tamaño de la cabecera.

El campo de ***checksum*** al igual que en IP, sirve como método de control de los datos, verificando que no han sido alterados. Este *checksum* cubre tanto la cabecera UDP como los datos enviados. Es necesario debido a que el *checksum* del protocolo IP tan sólo cubre la cabecera IP y no los datos que transporta. Si se detecta un error en el *checksum*, el datagrama es descartado sin ningún tipo de aviso.

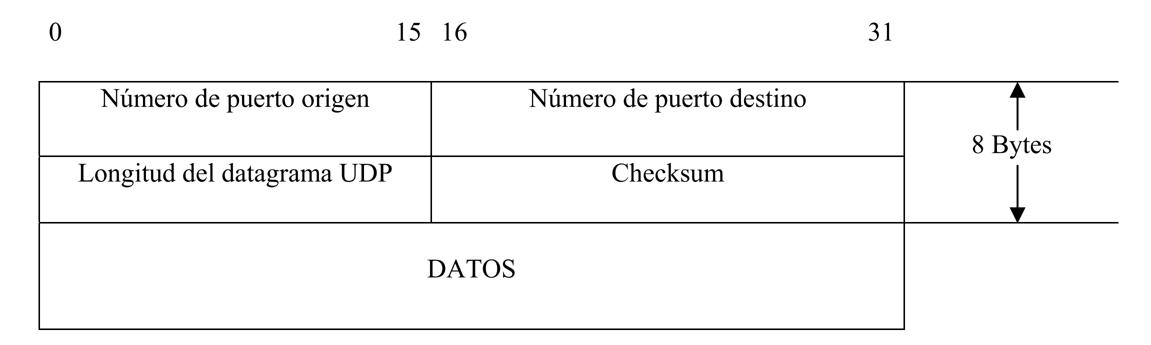


FIG. 1-10: Cabecera de un datagrama UDP.

1.4 El protocolo TCP

El protocolo TCP (*Transmission Control Protocol*) se podría definir como un protocolo **orientado a conexión**, **fiable** y **orientado a un flujo de bytes** [Ric98-1]. Su definición se recoge en **[RFC793]** publicado por Postel en 1981.

Aunque el protocolo TCP al igual que UDP utiliza los servicios de IP para su transporte por Internet (ver figura 1-11), es un protocolo **orientado a conexión**. Esto significa que las dos aplicaciones envueltas en la comunicación (usualmente un cliente y un servidor), deben establecer previamente una comunicación antes de poder intercambiar datos.

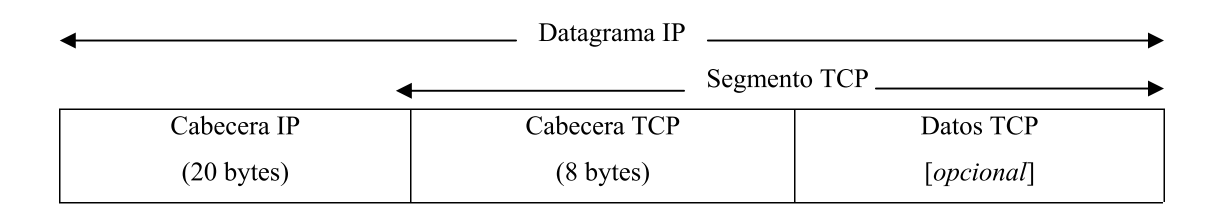


FIG. 1-11: Estructura de un segmento TCP.

TCP es también un protocolo **fiable**. La fiabilidad proporcionada por este protocolo viene dada principalmente por los siguientes aspectos:

1. Los datos a enviar son reagrupados por el protocolo en porciones denominadas *segmentos* [Ric98-1]. El tamaño de estos segmentos lo asigna el propio protocolo. Esto lo diferencia claramente de UDP, dónde cada porción de datos generada corresponde a un datagrama.

2. Cuando en una conexión TCP se recibe un segmento completo, el receptor envía una respuesta de confirmación (*Acknowledge*) al emisor confirmando el número de bytes correctos recibidos. De esta forma, el emisor da por correctos los bytes enviados y puede seguir enviando nuevos bytes.

3. Cuando se envía un segmento se inicializa un temporizador (*timer*). De esta forma, si en un determinado plazo de tiempo no se recibe una confirmación (*Acknowledge*) de los datos enviados, estos se retransmiten.

4. TCP incorpora un *checksum* para comprobar la validez de los datos recibidos. Si se recibe un segmento erróneo (fallo de *checksum* por ejemplo), no se envía una confirmación. De esta forma, el emisor retransmite los datos (bytes) otra vez.

5. Como IP no garantiza el orden de llegada de los datagramas, el protocolo TCP utiliza unos números de secuencia para asegurar la recepción en orden, evitando cambios de orden y/o duplicidades de los bytes recibidos.

6. TCP es un protocolo que implementa un control de flujo de datos. De esta forma, en el envío de datos se puede ajustar la cantidad de datos enviada en cada segmento, evitando colapsar al receptor. Este colapso sería posible si el emisor enviara datos sin esperar la confirmación de los bytes ya enviados.

La cabecera del segmento TCP (figura 1-12), es bastante más compleja que la de UDP debido a que la comunicación es más elaborada y debe proporcionar fiabilidad. Esto implica una serie de información adicional que debe mantenerse para poder conocer el estado de la comunicación en cualquier momento.

El número de **puerto origen** y número de **puerto destino**, sirven para diferenciar una comunicación en un ordenador de las demás. Cumple la misma función que en el datagrama UDP.

La tupla formada por la dirección IP y el número de puerto se denomina *socket*. Este término se utilizó luego en la especificación de la interface de programación de Berkeley (API).

Análogamente, la agrupación de las dos tuplas que definen una conexión entre dos ordenadores se denomina *socket pair*. En [WWW58] se puede obtener una lista completa de los servicios asociados a los puertos TCP y UDP.

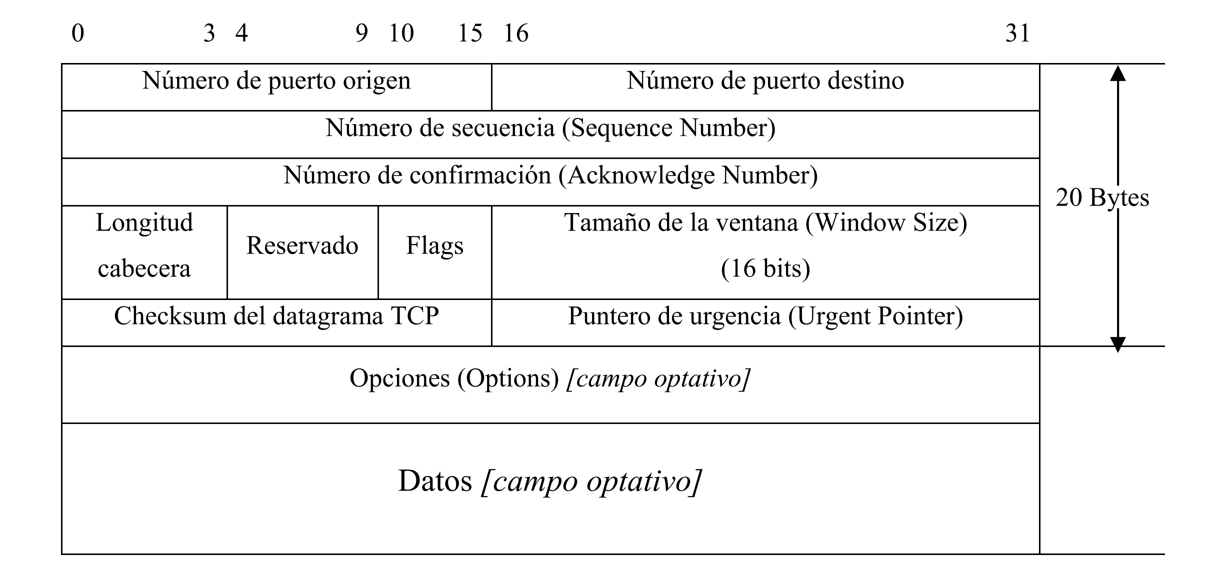


FIG. 1-12: Cabecera de un segmento TCP.

El **número de secuencia** (*Sequence Number*), identifica el byte concreto del flujo de datos que actualmente se envía del emisor al receptor. De esta forma, TCP numera los bytes de la comunicación de una forma consecutiva a partir del número de secuencia inicial. Cuando se establece una comunicación, emisor y receptor eligen un número de secuencia común, lo que nos permite implementar mecanismos de control como asegurar que los datos lleguen en el orden adecuado. Es un número de 32 bits, con lo que podemos enviar 2^32-1 bytes antes de que cicle.

El **número de confirmación** (*Acknowledge Number*), es el número de secuencia más uno. De este modo se especifica al emisor que los datos enviados hasta este número de secuencia menos uno son correctos. De aquí la importancia de la elección al principio de la comunicación de un número de secuencia común.

La **longitud de la cabecera** (*header Length*), especifica en palabras de 32 bits (4 bytes) el tamaño de la cabecera del segmento TCP incluyendo las posibles opciones. De esta forma el tamaño máximo es 15 * 4 = 60 bytes. No obstante, lo usual es tener un tamaño de 20 bytes (cabecera dónde no se incluyen opciones).

Las **banderas** (*Flags*), son las encargadas de especificar los diferentes estados de la comunicación. Así mismo, también validan los valores de los distintos campos de la cabecera de control. Puede haber simultáneamente varios *flags* activados. En la figura 1-13 podemos ver los distintos *flags* existentes y su significado.

FLAG	**Significado**
URG	Si es válido el puntero de urgencia.
ACK	El valor situado en el campo de confirmación (acknowledge) es válido.
PSH	El receptor debe pasar los datos a la aplicación o antes posible.
RST	RESET de la conexión
SYN	Inicio de comunicación. Búsqueda de un número de secuencia común.
FIN	El emisor finaliza el envío de datos.

FIG. 1-13: Flags del segmento TCP.

El **tamaño de la ventana** (*Window Size*), es el número de bytes desde el número especificado en el campo de confirmación, que el receptor está dispuesto a aceptar. El tamaño máximo es de (2^16) 65535 bytes. De esta forma, el protocolo TCP permite la regulación del flujo de datos entre el emisor y el receptor.

El **checksum del segmento** TCP, al igual que el del UDP o IP, tiene la función de controlar los posibles errores que se produzcan en la transmisión. Este *checksum* engloba la cabecera TCP y los datos. En caso de error, el datagrama/segmento queda descartado y el propio protocolo es el encargado de asegurar la retransmisión de los segmentos erróneos y/o perdidos.

El **puntero de urgencia** (*Urgent Poiner*), es válido sólo si el *flag* de **URG** se encuentra activado. Consiste en un valor positivo que se debe sumar al número de secuencia especificando una posición adelantada dónde podemos enviar datos urgentes.

Las **opciones** (*Options*), nos permiten especificar de forma opcional características extras a la comunicación. Un ejemplo de las opciones es el MSS (*Maximum Segment Size*), que especifica el tamaño máximo de datos que el emisor desea recibir [Ric98-1]. Esta opción se indica al inicio de la comunicación (*flag* SYN activado).

Los **datos** (*Data*) son opcionales. Esto significa que podemos enviar simplemente cabeceras TCP con diferentes opciones. Esta característica se utiliza por ejemplo al iniciar la comunicación o en el envío de confirmaciones. De esta manera, minimizamos el *overhead* ya que tan sólo enviamos/recibimos lo necesario para establecer o confirmar la comunicación.

1.5.1 Establecimiento de conexión TCP

TCP es un protocolo orientado a conexión. Esto implica que se ha de realizar un paso previo antes de poder intercambiar datos. Este paso es el de establecimiento de conexión.

Este paso es fundamental para poder garantizar la fiabilidad del protocolo, ya que es en este paso previo dónde se obtienen los números de secuencia que permitirán gestionar cualquier intercambio entre lo dos extremos de la comunicación. El método escogido para establecer la conexión se denomina **protocolo de 3 pasos** (*three way handshake,* ver figura 1-14).

En este esquema, podemos diferenciar un cliente activo que inicia la conexión (*active open*) y un servidor pasivo que tan sólo se limita a contestar (*passive open*) a las peticiones de conexión. Los tres pasos desarrollados en la petición de conexión son:

1. El cliente (ordenador que desea iniciar la comunicación) selecciona un número aleatorio de secuencia (***x*** en la figura 1-14). A continuación activa el *flag* de SYN en el campo de opciones. Finalmente envía un segmento TCP al ordenador destino.

2. El servidor (ordenador con el que se quiere establecer una comunicación) recibe la petición y almacena el número de secuencia ***x***. Elige un número aleatorio (***y*** en la figura 1-14) que utilizará como número de secuencia y activa los *flags* SYN y ACK.

   Finalmente envía un segmento con el número de secuencia elegido y con una confirmación del valor recibido mas uno (***ACK x+1*** en la figura 1-14).

3. El cliente almacena el número de secuencia (***y*** en la figura 1-14). Activa el *flag* de ACK y finalmente envía una confirmación del número recibido mas uno ***(ACK y+1*** en la figura 1-14).

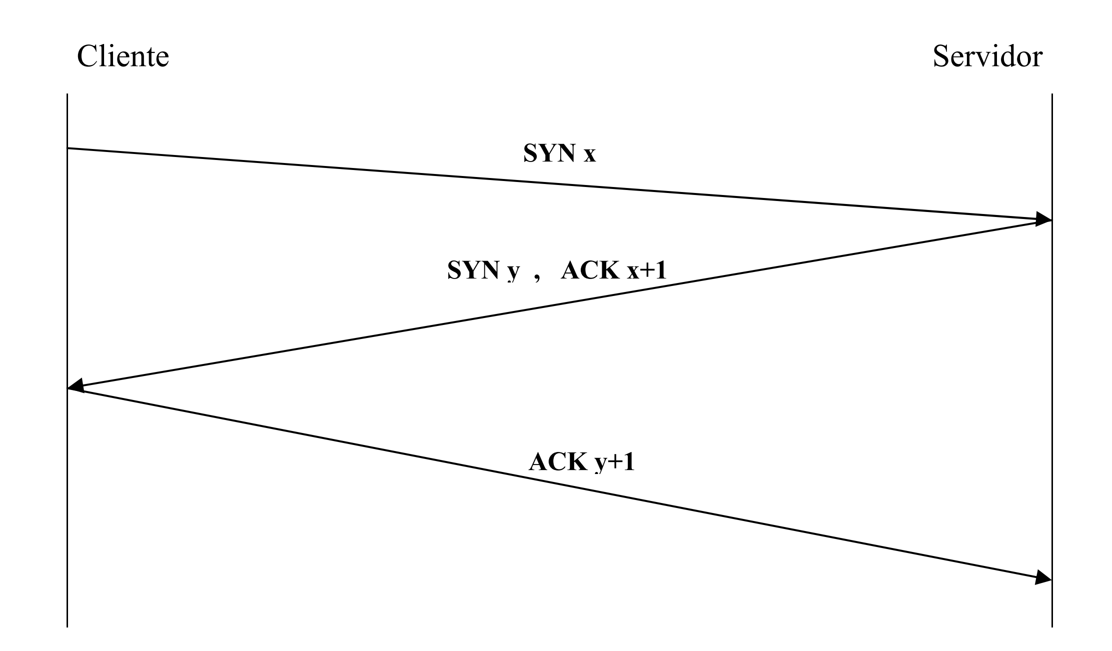


FIG. 1-14: Establecimiento de una conexión TCP (*three way handshake*).

En el caso de que este tercer paso no se realice, después de un cierto tiempo (entre 70 y 130 segundos dependiendo del sistema operativo) la conexión se liberará.

Esta característica se produce porque cuando se inicia una conexión TCP también se inicializa un temporizador (*timer*), que al finalizar (*time-out*) nos permite liberar la conexión y los recursos asociados.

1.5.2 Finalización de conexión TCP

Tal y como hemos visto en el punto anterior, se necesitan tres acciones para iniciar una conexión TCP. Para finalizarla se necesitan cuatro acciones debido a que la comunicación es *full duplex* (los datos pueden ser enviados y/o recibidos independientemente y en cualquier dirección de la comunicación).

Esto obliga a que cada sentido de la comunicación deba ser finalizado independientemente (ver figura 1-15).

Debido a la posibilidad de que cualquiera de los dos extremos implicados en la comunicación pueda enviar y/o recibir datos, tenemos la posibilidad de que cualquiera de los dos extremos finalice la comunicación (enviando una señal ***FIN***) hacia su sentido una vez finalizado el proceso de enviar/recibir datos. Esto nos obliga a realizar dos medias finalizaciones (*half-close*) de conexión, una hacia cada sentido.

TCP proporciona la capacidad de que cualquiera de los dos extremos de la conexión pueda finalizar su salida (*output*) de datos permitiéndole todavía recibir datos del otro extremo. Esta capacidad se denomina ***haf-close***.

En el caso de finalizar sólo un sentido de la comunicación (de cliente a servidor por ejemplo), el cliente puede seguir recibiendo datos del servidor enviando únicamente reconocimiento de datos (acknowledge, ***ACK***). De esta forma cuando el servidor envíe una señal de ***FIN*** la conexión TCP finalizará totalmente.

El envío se una señal ***FIN*** tan solo indica que no se producirá mas intercambio de datos en ese sentido. De esta forma es posible que en una conexión TCP se continúen enviando datos después de recibir una señal ***FIN***. Esta posibilidad es muy poco aprovechada en las aplicaciones que utilizan TCP actualmente.

En un proceso de *half-close* podemos diferenciar de forma clara un extremo activo (*active close*) y un extremo pasivo (*passive close*).

El extremo activo es el que envía la señal de FIN para finalizar ese sentido de la comunicación, mientras que el extremo pasivo se limita a aceptar la petición y enviar un reconocimiento (*acknowledge*) de final

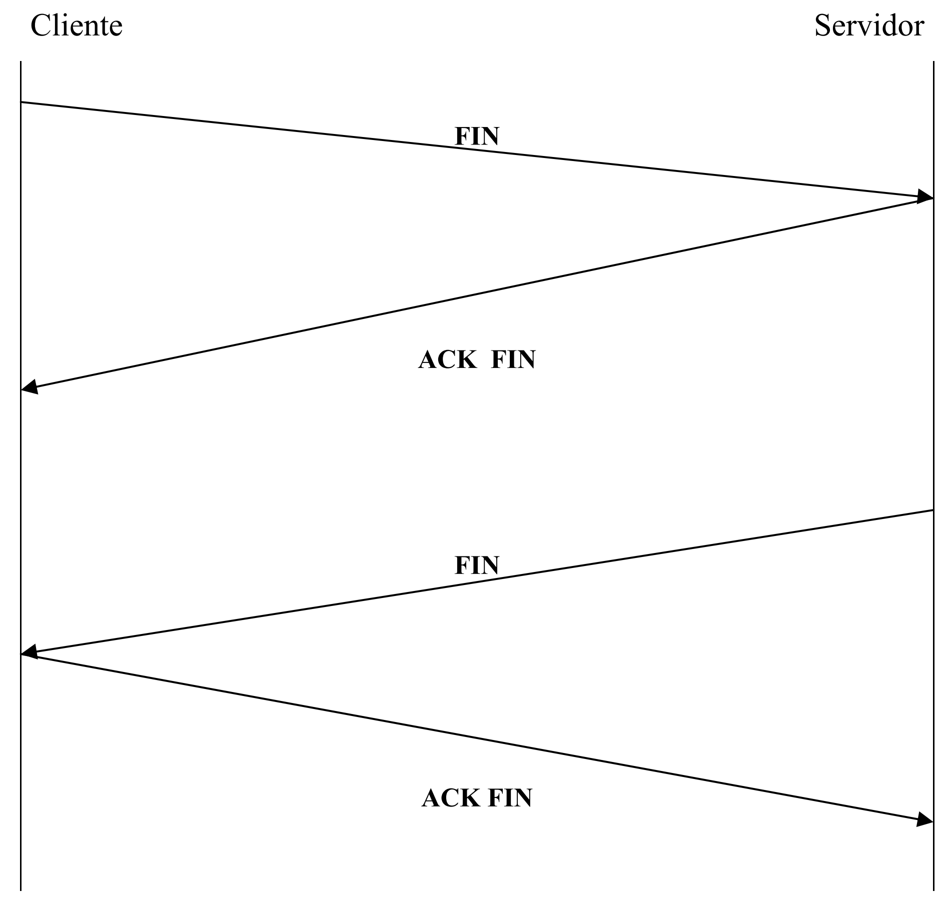


FIG. 1-15: Finalización de una conexión TCP.

De esta forma tenemos que para finalizar una conexión TCP debemos finalizarla en cada uno de los dos sentidos. Esto obliga a que alternativamente el cliente y el servidor adopten los papeles activo y pasivo en un proceso que consta de 4 fases:

1. (El cliente adopta el papel activo) El cliente decide finalizar la comunicación en su sentido. Enviando al servidor una señal de finalización (***FIN)*** con un número de secuencia.

2. (El servidor adopta el papel pasivo) El servidor recibe esta señal y responde con un reconocimiento (*acknowledge, ACK*) de señal. Enviando el número de secuencia recibido más uno.

   Cabe señalar que la finalización (FIN) al igual que la señal de petición de conexión (***SYN***) consume un número de secuencia. Finalización de la primera *half-close*.

3. (El servidor adopta un papel activo) El servidor decide finalizar la conexión en su sentido y envía una señal de finalización (***FIN***) de conexión al cliente.

4. (El cliente adopta un papel pasivo) El cliente acepta la petición de finalizar la conexión respondiendo con un ***ACK*** y enviando el número de secuencia recibido mas uno. Finalización de la segunda *half-close*. Finalización de la conexión TCP.

1.6 Encaminamiento de datagramas

El proceso de rutado o encaminamiento (*routing*) hace referencia a cómo los distintos datagramas IP enviados circulan y seleccionan el camino hacia su destino.

Debido a la construcción de Internet para que fuera tolerante a fallos, estos caminos no son fijos o estáticos, sino que existen diferentes vías que se actualizan dinámicamente para que un paquete alcance su destino sin necesidad de usar siempre la misma ruta. De esta forma, tenemos que cada paquete se encamina de forma independiente hacia su destino.

Los ordenadores encargados de recibir y distribuir los datagramas hasta su siguiente paso hacia el destino se denominan *routers*. Estos routers se encargan de leer la dirección destino del paquete y seleccionar su vía de salida.

En la figura 1-16 podemos ver el proceso de encaminamiento, dónde un router recibe un datagrama IP y lo conduce hasta el siguiente router.

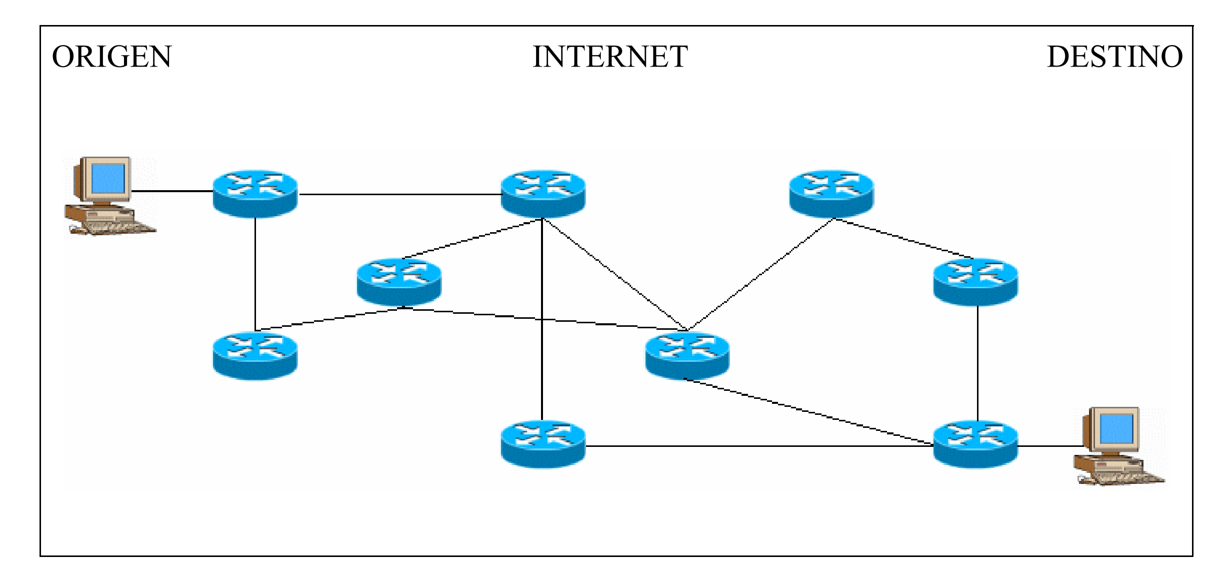


FIG. 1-16: Rutado de paquetes en Internet.

Podemos ver pues que esta comunicación se produce mediante saltos unitarios (*hops*) de router a router hasta que alcanza su destino o el paquete es descartado (ya sea porque no exista una ruta al destino de la información, o porque se ha consumido el tiempo de vida del datagrama).

Una vez que el datagrama llega hasta el router final, este se propaga por la red local hasta su destino. A diferencia que de lo que ocurre en Internet, en las redes locales (LAN) el direccionamiento de los ordenadores conectados se produce mediante una dirección hardware única denominada dirección **MAC** (*Media Access Control*) [WWW25][WWW26][WWW32].

La dirección MAC es única para cada dispositivo de red y viene serigrafiada en el hardware de la tarjeta, consiste en un número de 48 bits que indican el fabricante y su número de serie (ver figura 1-17).

Esta dirección suele expresarse como 12 dígitos hexadecimales (0-F) que se dividen en 24 bits (6 dígitos hexadecimales) para el número de organización OUI (*Organiztion Unique Identifier*) y 24 bits para la dirección de número de serie.

Una vez que el router final recibe un paquete IP para un ordenador que está conectado en su red local (LAN), realiza una petición a toda la red local para que se identifique el ordenador destinatario y así poder enviarle el datagrama IP.

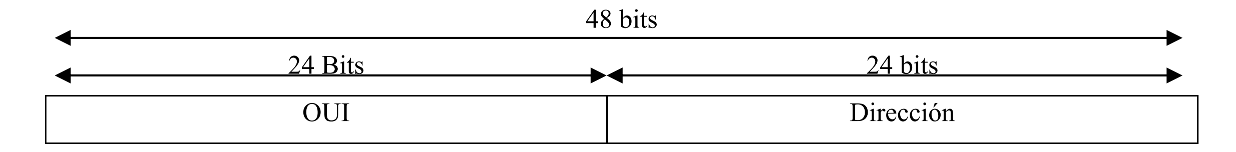


FIG. 1-17: Dirección MAC.

Para facilitar este proceso, existen dos protocolos para la obtención de la dirección MAC de una dirección IP dada (***ARP***) y otro para la obtención de la dirección IP de una dirección MAC (***RARP***) [WWW12][WWW13][RFC903].

- **ARP** (*Address Resolution Protocol*) [WWW27]: Obtención de la dirección MAC de una dirección IP. Al recibir un paquete IP para X.Y.Z.T, el router pregunta cual es la dirección MAC de esta dirección IP para enviarla por la red local.

- **RARP** (*Reverse ARP*) [WWW28]: Obtención de la dirección IP correspondiente a una dirección MAC dada. Usado por ejemplo en las máquinas que usan DCHP para la obtención de una dirección IP automática.

El uso de los protocolos ARP y RARP propaga mensajes a todos los ordenadores conectados en la red local, ya que el destinatario debe identificarse de entre todos los posibles candidatos. Los ordenadores que no responden ignoran estas peticiones.

En el protocolo IP también existe una forma de identificar la red a la que pertenece la dirección IP, para ello simplemente debemos sustituir los bits de la máscara de red por ceros (ver capítulo de redes IP).

Análogamente, IP también tiene un sistema de radiodifusión (*broadcast*) [RFC919] que permite la comunicación simultánea con **todos** los ordenadores de la misma red, simplemente debemos sustituir los bits de la máscara de red por unos.

1.7 Resumen

En este capítulo se ha realizado una presentación de la familia de protocolos TCP/IP versión 4 que abarca tanto sus definiciones formales como sus aspectos funcionales.

IP (*Internet Protocol*), que es un protocolo **no fiable** y **no orientado a conexión** que se encarga del transporte de los datos por Internet.

UDP (*User Datagram Protocol*), protocolo **simple** y **orientado a datagrama** que se encarga de enviar datagramas usando los servicios del protocolo IP.

ICMP (*Internet Control Message Protocol*), protocolo encapsulado en datagramas IP que se encarga del control del flujo de la comunicación y de los errores que puedan ocurrir.

TCP (*Transmission Control Protocol*) que es un protocolo **fiable**, **orientado a conexión** y **orientado a byte** que utiliza los servicios del protocolo IP. Este protocolo establece una conexión previa con el destino (mediante el *three way handshake*) que le permite regular el flujo de bytes enviados y su correcta recepción, retransmitiendo los bytes recibidos incorrectamente o perdidos. Finalmente realiza una fase de desconexión del receptor (mediante el sistema de *half close*) antes de dar por concluida la comunicación.

También se ha comentado el proceso de encaminamiento o rutado (routing) de datagramas por Internet así como su tránsito final hacia la red local (LAN) dónde alcanza al ordenador de destino mediante el uso de ARP y RARP.

CAPITULO 2

Denegación de servicio: DOS / DDOS

En este capítulo se realizará un análisis detallado de los ataques de denegación de servicio. Estos ataques están destinados a eliminar total o parcialmente la presencia en Internet de un ordenador o servicio.

Primero ubicaremos históricamente el contexto tecnológico y luego pasaremos a las distintas motivaciones de los atacantes para ir comentando la evolución y los diferentes tipos de ataques de denegación de servicio:

- Ataques de denegación de servicio simples (*Deny Of Service*). Este tipo de ataques se caracterizan por tener un único origen desde el cual se realiza la denegación del servicio.

- Ataques de denegación de servicio distribuido (*Distributed DOS o DDOS*). En este tipo de ataques se utilizan varias fuentes coordinadas que pueden realizar un ataque progresivo, rotatorio o total. Realizaremos un estudio detallado de las distintas herramientas que los componen, sus arquitecturas y sus tácticas más comunes.

Finalmente realizaremos una exposición de un ataque DDOS real al un servidor WWW, describiendo las técnicas que se emplearon así como los efectos conseguidos durante el tiempo de duración de este ataque.

2.1 Contexto histórico y tecnológico

En la década de los noventa tras el gran apogeo de Internet potenciado en gran medida por el WWW, los ataques "clásicos" de un hacker que conseguía el número de teléfono de un modem conectado a un ordenador pasan a la historia.

La conectividad del mundo está asegurada por Internet y cualquier ordenador conectado puede convertirse en blanco de los ataques de cualquier otro usuario conectado a la red de redes. La difusión de los protocolos que gobiernan Internet (IP, TCP, UDP, ICMP...) permiten que a mediados de los noventa se empiecen a registrar los primeros ataques por red (1995/1996). Estos ataques se solían utilizar en las guerras de IRC dónde el objetivo era desconectar al contrincante para conseguir el puesto de operador de un canal [WWW114].

Con la mejora de las comunicaciones y los aumentos de anchos de banda disponibles para los usuarios conectados (modems de 33k/56k/RDSI...) estos ataques van adaptándose a los recursos disponibles por los atacantes, empezando a hacer uso extensivo de los anchos de banda existentes con el objetivo de saturar la conexión de la víctima.

En 1997 aparece la primera herramienta efectiva para la realización de ataques de denegación de servicio (TRIN00). Posteriormente aparecen cientos de derivados y mejoras de esta herramienta (1998 TFN, 1999 TFN2K...) que Irán ampliando las características utilizadas en los ataques.

La gran repercusión en los medios de comunicación masiva se produce cuando estas herramientas empiezan a ser utilizadas contra servicios de empresas internacionales (Yahoo, Ebay, Microsoft...) y gobiernos de todo el mundo.

En la actualidad, la evolución de estas herramientas de denegación de servicio va ligada al desarrollo de Internet. Los aumentos espectaculares de anchos de banda (ADSL, Frame Relay, ATM, OC-xx…) ha dado lugar a que la capacidad necesaria para la saturación de una comunicación sea mayor, con lo que un sólo nodo puede no ser necesario para conseguir el objetivo.

De esta forma, los ataques combinados o distribuidos empiezan a aparecer a finales del 2000 y en la actualidad son los preferidos por los hackers.

2.2 Fuentes de origen de los ataques DOS / DDOS

Los ataques de denegación de servicio suelen tener varios orígenes, lo que complica la posibilidad de mantener un control efectivo sobre todos los recursos o servicios ofrecidos. No obstante, los distintos orígenes pueden agruparse en:

1. **Usuarios legítimos o internos***:* Este grupo se subdivide en aquellos usuarios poco cuidadosos que colapsan el sistema o servicio inconscientemente (por ejemplo la persona que llena el disco duro del sistema bajando archivos de música), usuarios malintencionados (aquellos que aprovechan su acceso para causar problemas de forma premeditada) y usuarios ladrones que utilizan el acceso de un usuario legítimo (ya sea robándolo del usuario legítimo o aprovechándose de la confianza de este).

2. **Agentes externos***:* Este grupo hace referencia a los no usuarios del sistema. De esta forma se consigue acceso al recurso o servicio sin necesidad de ser un usuario legítimo (un sistema que no controle la autenticación de usuarios, un sistema que presente un "*bug*" conocido...). En este grupo usualmente se falsea la dirección de origen (*faked/spoofed IP*) con el propósito de evitar el origen real del ataque.

Además, cabe recalcar que gran parte de la peligrosidad de este tipo de ataques por red viene dada por su independencia de plataforma hardware o sistema operativo.

Debido a que el protocolo IP permite una comunicación homogénea (independiente del tipo de ordenador o fabricante) a través de espacios heterogéneos (redes ETHERNET, ATM...), un ataque exitoso contra el protocolo IP se convierte inmediatamente en una amenaza real para todos los ordenadores conectados a Internet.

2.3 DOS

Definiremos la **denegación de servicio** (*Deny Of Service, DOS*) como la imposibilidad de acceso a un recurso o servicio por parte de un usuario legítimo.

De esta forma podemos definir un **ataque de denegación de servicio** (*DOS Attack*) como *"la apropiación exclusiva de un recurso o servicio con la intención de evitar cualquier acceso de terceros. También se incluyen en esta definición los ataques destinados a colapsar un recurso o sistema con la intención de destruir el servicio o recurso*" [WWW45].

Una definición más restrictiva de los ataques de denegación de servicio en redes IP podemos encontrarla en la bibliografía [WWW10] [WWW11] dónde se define como la consecución total o parcial (temporal o totalmente) del cese en la prestación de servicio de un ordenador conectado a Internet.

A continuación realizaremos una exposición de los ataques de denegación de servicio (DOS) mas comunes y que han sido registrados por organismos internacionales y grupos de investigación [Nor99] de todo el mundo como el CERT [WWW21].

2.3.1 IP Flooding

El ataque de IP Flooding (inundación de paquetes IP) se realiza habitualmente en redes locales o en conexiones con un gran ancho de banda disponible.

Consiste en la generación de tráfico espurio con el objetivo de conseguir la degradación del servicio de red. De esta forma, el atacante consigue un gran consumo del ancho de banda disponible ralentizando las comunicaciones existentes en toda la red.

Se da principalmente en redes locales dónde el control de acceso al medio es nulo y cualquier máquina puede enviar/recibir sin ningún tipo de limitación en el ancho de banda consumido (ver figura 2-1).

El tráfico generado en este tipo de ataque puede ser:

- **Aleatorio:** Cuando la dirección de origen o destino del paquete IP es ficticia o falsa. Este tipo de ataque es el más básico y simplemente busca degradar el servicio de comunicación del segmento de red dónde el ordenador responsable del ataque está conectado.

- **Definido o dirigido:** Cuando la dirección de origen, destino (o ambas) es la de la máquina que recibe el ataque. El objetivo de este ataque es doble, ya que además del colapso del servicio de red dónde el atacante genera los paquetes IP busca colapsar un ordenador destino, ya sea reduciendo el ancho de banda disponible para que siga ofreciendo el servicio o colapsar el servicio ante una gran cantidad de peticiones que el servidor será incapaz de procesar.

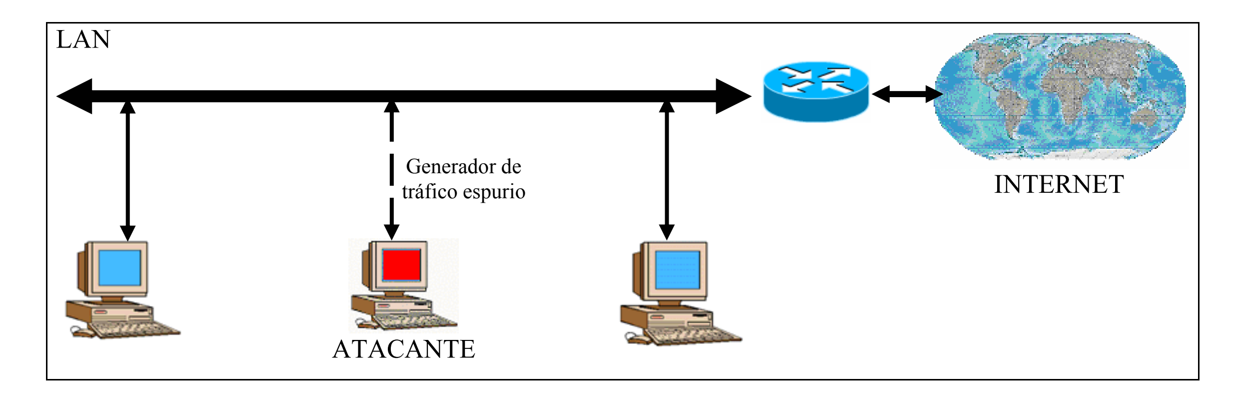


FIG. 2-1: Ataque IP Flooding.

Este tipo de ataque por inundación se basa en la generación de datagramas IP de forma masiva. Estos datagramas pueden ser de los tipos siguientes (ver capítulo 1 y [Ric98-1]):

- **UDP**: Generar peticiones sin conexión a cualquiera de los 65535 puertos disponibles. En muchos sistemas operativos, las peticiones masivas a puertos específicos UDP (ECHO, WINS...) causan el colapso de los servicios que lo soportan.

- **ICMP**: Generación de mensajes de error o control de flujo malicioso. En este caso el objetivo es doble, degradar el servicio de red con la inundación de peticiones y/o conseguir que los sistemas receptores quede inutilizados por no poder procesar todas las peticiones que les llegan.

- **TCP**: Genera peticiones de conexión con el objetivo de saturar los recursos de red de la máquina atacada. Este protocolo es orientado a conexión, y como tal consume recursos de memoria y CPU por cada conexión. El objetivo es el de saturar los recursos de red disponibles de los ordenadores que reciben las peticiones de conexión y degradar la calidad del servicio.

2.3.2 Broadcast

En el protocolo IP también existe una forma de identificar la red a la que pertenece la dirección IP, para ello simplemente debemos sustituir los bits de la máscara de red por ceros (ver capítulo 1).

Análogamente, IP también tiene un sistema de radiodifusión (*broadcast*) [RFC919] que permite la comunicación simultánea con **todos** los ordenadores de la misma red. Para realizar esta operación simplemente debemos sustituir los bits de la máscara de red por unos.

En este tipo de ataque se utiliza la dirección de identificación de la red IP (*broadcast address*) como dirección de destino del paquete IP. De esta forma, el router se ve obligado a enviar el paquete a todos los ordenadores pertenecientes a la red, consumiendo ancho de banda y degradando el rendimiento del servicio.

También existen variantes dónde se envían peticiones de PING a varios ordenadores falseando la dirección IP de origen y substituyéndola por la dirección de *broadcast* de la red a atacar. De esta forma, todas las respuestas individuales (pong en la figura 2-2) se ven amplificadas y propagadas a todos los ordenadores pertenecientes a la red.

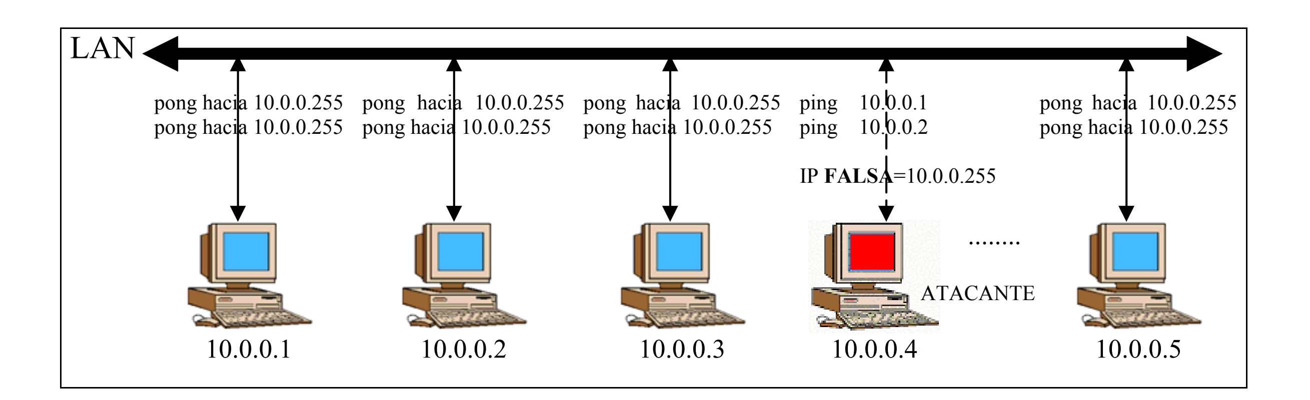


FIG. 2-2: Ataque Broadcast.

Este ataque al igual que el IP Flooding se suele realizar en redes locales ya que requiere un gran ancho de banda por parte del atacante, aunque con la mejora de las comunicaciones y anchos de banda disponibles es factible realizarlo remotamente.

2.3.3 Smurf

El protocolo ICMP [RFC792] es el encargado de realizar el control de flujo de los datagramas IP que circulan por Internet. Este protocolo consta de diversas funcionalidades que permiten desde la comunicación de situaciones anómalas (no se ha podido realizar la entrega del paquete IP) hasta la comprobación del estado de una máquina en Internet (ping - pong o ECHO - ECHO REPLY).

Este tipo de ataque se basa en falsear las direcciones de origen y destino de una petición ICMP de ECHO (ping) (ver figura 2-3).

Como dirección de origen colocamos la dirección IP de la máquina que va a ser atacada. En el campo de la dirección de destino situamos la dirección *broadcast* de la red local o red que utilizaremos como "lanzadera" para colapsar al sistema elegido.

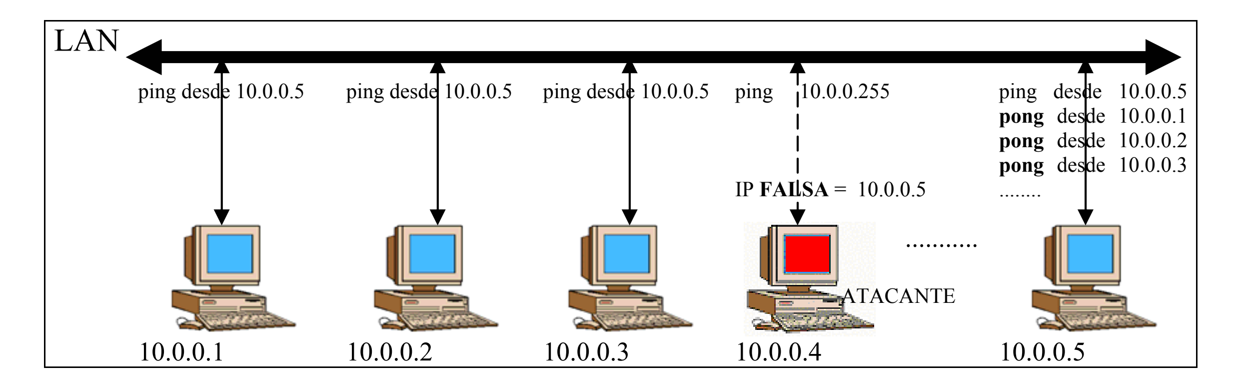


FIG. 2-3: Ataque Smurf.

Con esta petición fraudulenta, se consigue que todas las máquinas de la red contesten a la vez a una misma máquina, consumiendo el ancho de banda disponible y saturando al ordenador elegido [WWW14] [WWW15].

2.3.4 Teardrop / Boink / Bonk / Nestea

Por definición del protocolo, un paquete IP tiene un tamaño máximo de 64K (65.535 Bytes). De esta forma, si deseamos enviar una cantidad mayor de datos, deberemos enviar varios paquetes IP al destino.

Análogamente, cada red por la que transitan los paquetes IP tiene un tamaño máximo de paquete (**MTU**[ℜ], *Maximum Transfer Unit*), por lo que necesitaremos fragmentar los paquetes IP en varios trozos que serán reconstruidos al llegar al destino.

Para tratar el tema de la fragmentación el protocolo IP especifica unos campos en la cabecera encargados de señalar si el paquete IP está fragmentado (forma parte de un paquete mayor) y que posición ocupa dentro del datagrama original.

En el campo de banderas (*flags*) existe un bit denominado "*More*" (mas) que indica que el paquete recibido es un fragmento de un datagrama mayor, igualmente el *campo de número de identificación* del datagrama especifica la posición del fragmento en el datagrama original (ver capítulo 1 para más información).

El ataque "*teardrop*" [WWW16] utiliza esta funcionalidad para intentar confundir al sistema operativo en la reconstrucción del datagrama original y lograr el colapso del servicio y/o del sistema:

[ℜ] En el caso de redes ETHERNET es de 1.500 bytes.

- Supongamos que deseamos enviar un fichero de 1024 bytes en una red con un MTU de 512 bytes. Enviando dos fragmentos de 512 bytes tenemos suficiente.

	POSICIÓN	**LONGITUD**
Fragmento 1	0	512
Fragmento 2	512	512

FIG. 2-4: Fragmentación correcta.

- Sin embargo, puedo realizar unas modificaciones en los campos de posición y longitud que introduzcan inconsistencias e incoherencias en la reconstrucción del paquete original.

	POSICIÓN	**LONGITUD**
Fragmento 1	0	512
Fragmento 2	500	512
....	....	....
Fragmento N	10	100

FIG. 2-5: Fragmentación **INCORRECTA**.

El ataque "*teardrop*" y sus variantes se basan en falsear los datos de posición y/o longitud de forma que el datagrama se sobrescriba (*overlapping*) y produzca un error de sobreescritura del buffer (*buffer-overrun*) al tratar con desplazamientos negativos.

Usualmente el sistema operativo detecta el intento de acceso a una zona de memoria que no corresponde al proceso ejecutado y aborta el servicio. Dependiendo de la robustez del sistema operativo podemos perder "solo" el servicio atacado o incluso llegar a colapsar todo el ordenador.

Otra variante de este ataque consiste en enviar cientos de fragmentos "modificados" de forma que se solapen con el objetivo de saturar la pila de protocolo IP de la máquina atacada.

2.3.5 ECHO-CHARGEN / Snork

Como la mayoría de protocolos, IP define un sistema de pruebas simple que permite verificar el funcionamiento del protocolo de comunicación. El sistema proporcionado se basa en enviar un datagrama "especial" al ordenador destino, que lo reconoce y envía una respuesta al origen (ECHO → REPLY).

El protocolo IP define para estas pruebas [RFC862] un servicio para la recepción de un datagrama UDP al puerto 7 (ECHO).

Por otro lado, existe un servicio proporcionado en muchos sistemas operativos tipo UNIX denominado **CHARGEN** (*CHARacter GENerator*, generador de caracteres) que dada una petición responde con una secuencia aleatoria de caracteres.

Este servicio se encuentra disponible "escuchando" en el puerto 19 a datagramas UDP. En sistemas Windows NT se suele utilizar el puerto 135 (*Microsoft Locator Service*) para el ataque "*snork*" [WWW43].

El ataque consiste en cruzar ambos servicios enviando una petición falsa al servicio CHARGEN (que retorna una secuencia de caracteres pseudo-aleatoria) falseando la dirección de origen dando como puerto de respuesta el puerto ECHO (que responde a cualquier petición) de la máquina a atacar. De esta forma, se inicia un juego de ping-pong infinito (ver figura 2-6).

Este ataque puede realizarse entre varios ordenadores (consumiendo ancho de banda y degradando el rendimiento de la red) o desde un mismo ordenador (él mismo se envía una petición y responde) consiguiendo consumir los recursos existentes (especialmente CPU y memoria) de la máquina atacada.

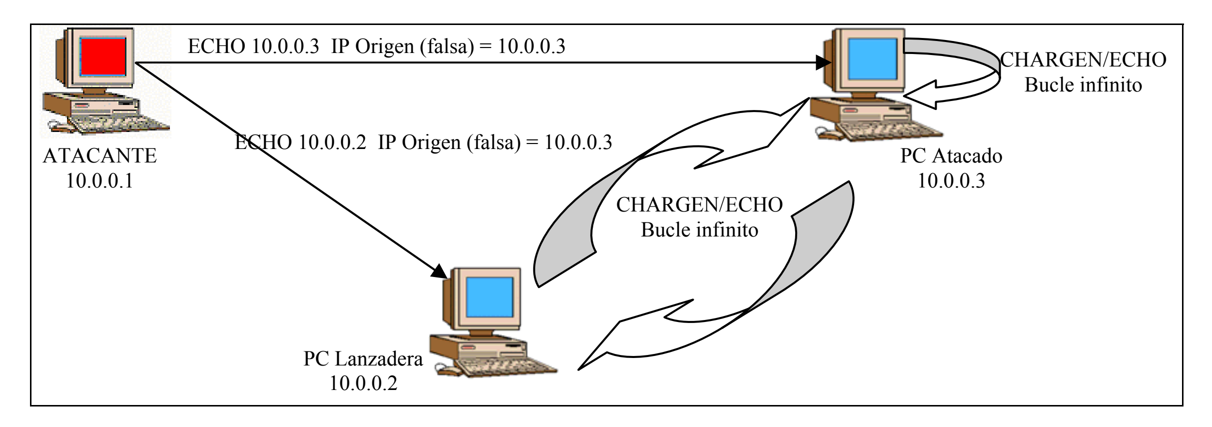


FIG. 2-6: Ataque CHARGEN-ECHO.

2.3.6 DOOM / QUAKE

Este ataque consiste en una generalización del ataque ECHO-CHARGEN, ya que se basa en buscar algún servicio activo (los servidores de juegos en red del DOOM y el QUAKE por ejemplo) que responda a cualquier datagrama recibido.

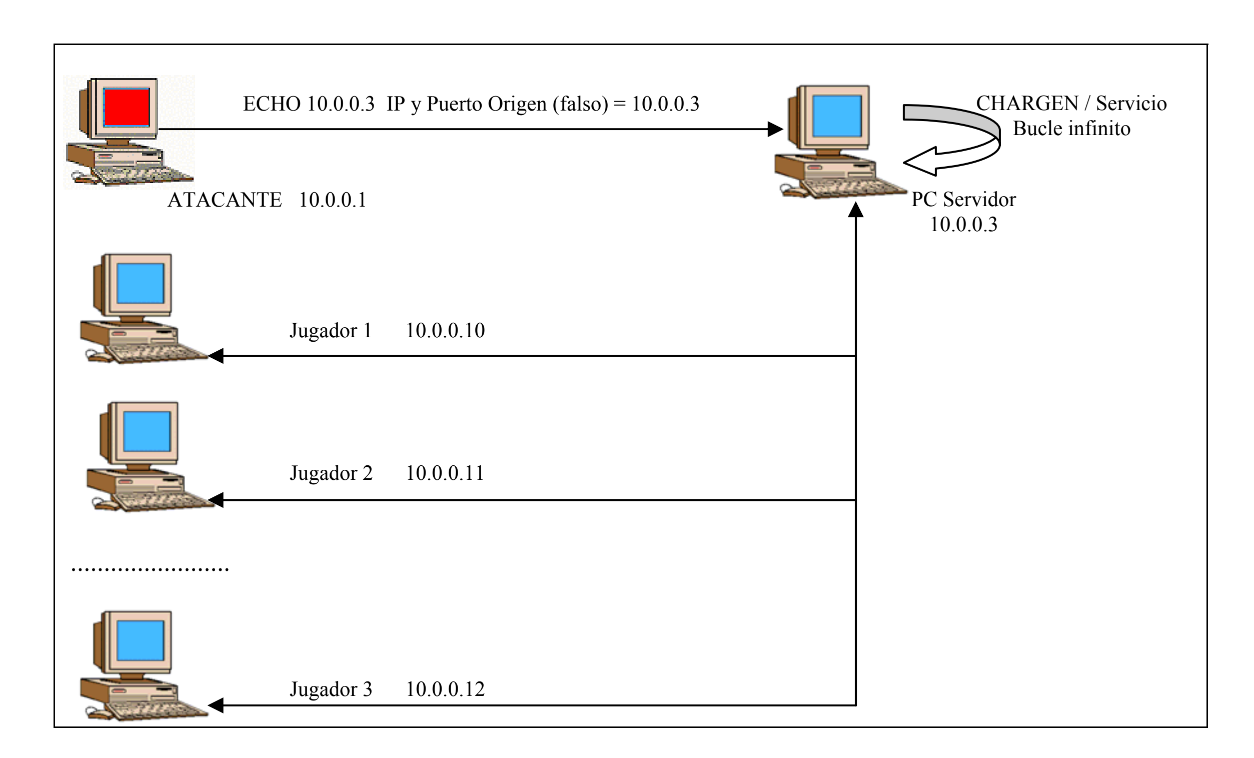


FIG. 2-7: Ataque CHARGEN-ECHO.

De esta forma, se envían peticiones dando como origen el puerto CHARGEN, lo que inicia un juego de petición/respuesta (ping-pong) infinito.

El objetivo en este caso es doble, ya que además de consumir recursos e intentar bloquear la máquina servidora del juego, el juego en cuestión queda también afectado (ver figura 2-7).

2.3.7 LAND

Este tipo de ataque se basa en falsear la dirección y puerto origen para que sean las mismas que la del destino. De esta forma, se envían al ordenador atacado peticiones de conexión desde él mismo hasta él mismo (ver figura 2-8).

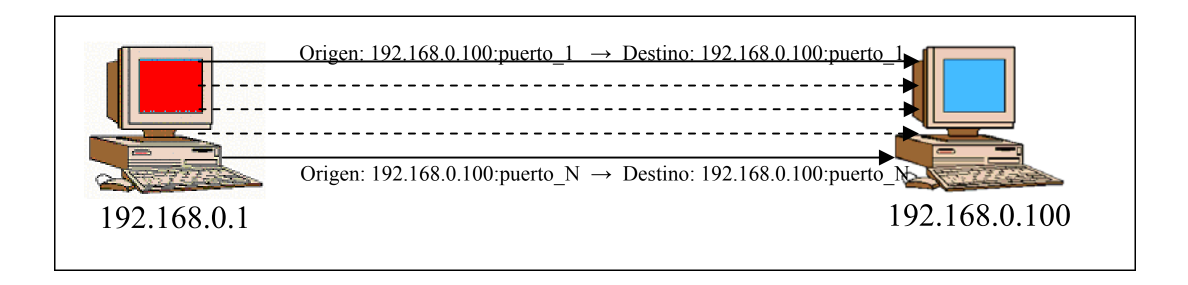


FIG. 2-8: Ataque LAND.

La mala calidad del software encargado de gestionar las comunicaciones en los sistemas operativos hace que muchas veces este sencillo ataque consiga colapsar el sistema atacado.

Usualmente este tipo de peticiones suelen ir acompañadas de violaciones expresas de los campos de opciones de los protocolos con el objetivo de confundir al ordenador atacado.

2.3.8 PING OF DEATH

El PING de la muerte (*Ping of death*) ha sido probablemente el ataque de denegación de servicio mas conocido y que mas artículos de prensa ha conseguido.

Este ataque utiliza una vez mas las definiciones de la longitud máxima de paquetes de los protocolos IP/UDP/TCP/ICMP [RFC791][RFC792] así como la capacidad de fragmentación de los datagramas IP.

La longitud máxima de un datagrama IP es de 64K (65535 Bytes) incluyendo la cabecera del paquete (20 Bytes) y asumiendo que no hay opciones especiales especificadas[ℜ].

El protocolo ICMP es el que se utiliza para la comunicación de mensajes de control de flujo en las comunicaciones (si la red está congestionada, si la dirección de destino no existe o es inalcanzable...) y tiene una cabecera de 8 bytes.

De esta forma tenemos que para enviar un mensaje ICMP tenemos disponibles 65535 - 20 - 8 = **65507 Bytes**.

Como se ha explicado en puntos anteriores, en el caso de enviar más de 65535 bytes el paquete se fragmenta y se reconstruye en el destino utilizando un mecanismo de posición y desplazamiento relativo.

No obstante, si enviamos ordenes al sistema operativo para que envíe un datagrama con una longitud de 65510 bytes (correcto, puesto que es inferior a 65507 bytes):

ping -l 65510 direcion_ip [Windows]
ping -s 65510 direccion_ip [Unix]

[ℜ] Ver capitulo 1 para mas detalles.

Obtenemos que el tamaño es inferior a los 65535 con lo que los datos a enviar cogen en un único paquete IP (fragmentado en N trozos, pero pertenecientes al mismo datagrama IP).

Sumando el tamaño de las cabeceras obtenemos:

20 bytes cabecera IP + 8 bytes cabecera ICMP + 65510 bytes de datos = **65538**!!!!!

Sin embargo debido a la cabecera ICMP el espacio disponible tan sólo era de 65507 bytes!!. En consecuencia al reensamblar el paquete en el destino se suelen producir errores de *overflow/coredump* que causan la parada del servicio o del sistema atacado.

2.4 DDOS

Podemos definir los ataques de denegación de servicio distribuido (DDOS) como un ataque de denegación de servicio (DOS) dónde existen múltiples focos distribuidos y sincronizados que focalizan su ataque en un mismo destino [Dit99][WWW46] (ver figura 2-9).

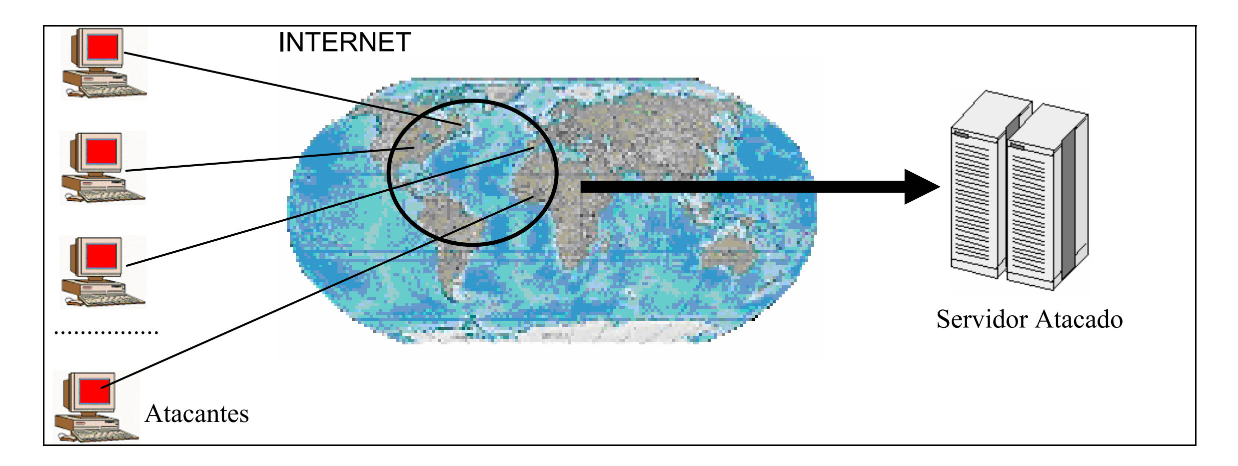


FIG. 2-9: Ataque típico de DDOS.

A continuación se presenta un análisis de los ataques de denegación de servicio distribuido (DDOS) más conocidos actualmente, se profundiza en el modelo distribuido de las fuentes del ataque así como su sincronización y *modus operandi* [WWW42].

Los análisis de las diferentes herramientas se presentan históricamente [ZDD02] para poder comparar la evolución en el diseño de los ataques DDOS.

2.4.1 TRINOO / TRIN00

El proyecto TRINOO [Dit99] también conocido como TRIN00, es una herramienta que implementa un ataque de denegación de servicio mediante un modelo jerárquico maestro/esclavo (*master*/*slave*).

Originariamente se encontró en sistemas Solaris desde dónde se produjeron los primeros ataques conocidos [WWW47] probablemente aprovechando bugs conocidos del sistema operativo (*buffer overflow*...). Una vez que los hackers obtuvieron privilegios de administrador pudieron instalar los programas y demonios (*daemons*) necesarios.

Una vez obtenido acceso a un ordenador, se procede a la instalación/compilación de todos los programas del proyecto TRINOO (*sniffers* de red, puertas traseras o *backdoors*, *daemons*, *root-kits*...).

Estos ordenadores suelen residir en grandes corporaciones o universidades dónde los anchos de banda de las comunicaciones son grandes y pueden pasar desapercibidos entre cientos o miles de ordenadores pertenecientes a la misma red.

Posteriormente, desde este ordenador comprometido se procede al rastreo (*scanning*) de otros ordenadores con vulnerabilidades conocidas en servicios básicos (FTP, RPC, NFS…) para proceder a su infección. Este rastreo suele realizarse dentro de la misma red ya que la velocidad de transmisión es más rápida y permite "verificar" más

máquinas por segundo. Por otro lado las medidas de seguridad entre ordenadores locales suelen ser mínimas.

No obstante, también se suelen lanzar rastreos a máquinas de redes externas siempre con mesura y procurando pasar desapercibidos entre el resto de tráfico normal de la red.

Con los resultados obtenidos del rastreo se genera una lista de ordenadores vulnerables dónde se ejecutarán los programas para obtener el acceso (*exploits*).

Para verificar que ordenadores de la lista han podido ser captados, el ordenador de origen suele tener un proceso demonio (*daemon*) que escucha el puerto TCP 1524, dónde se enviará una señal por cada ordenador infectado.

Otra variante es que cada máquina infectada envíe un e-mail a una cuenta gratuita. De esta forma el hacker manualmente introduce en una lista los ordenadores que tiene bajo su control (esta forma deja un rastro que puede ser seguido por la policía).

Es importante conocer que arquitectura y sistema operativo está presente en cada una de las máquinas infectadas, ya que el primer ordenador (origen) se encargará de distribuir los ficheros ejecutables (ya compilados!) a cada una de las máquinas infectadas a través de Internet.

El *script* (ver figura 2-10) deja un proceso planificado en el sistema CRON de la máquina que "escucha". De esta forma, este sistema se encarga de la ejecución de programas periódicos (compresión de los ficheros de logs del sistema...)

Hay que señalar que en este sistema necesitamos tener en la primera máquina (origen) las versiones compiladas de los diferentes programas. Pese a que este comportamiento parezca un defecto no lo es, ya que por un lado no todas las máquinas UNIX poseen compiladores instalados y/o no es posible compilar aplicaciones sin levantar sospechas.

Por un lado los compiladotes son grandes consumidores de recursos, memoria y CPU y por otro aseguramos que el código ejecutado sea el compilado por nosotros.

Con este sistema es realmente flexible vemos como a partir de un ordenador podemos llegar a obtener toda una red de máquinas a nuestra disposición. A veces los atacantes suelen instalar aplicaciones para esconder/falsear la presencia de estos programas (procesos) en el sistema [Dit02], especialmente en el ordenador de origen.

- Para conocer la arquitectura/plataforma y sistema operativo de un ordenador con Unix, existe el comando 'uname -a'

```
# uname -a
CYGWIN_NT-5.0 FIJO 1.3.12(0.54/3/2) 2002-07-06 02:16 i586 unknown
```

- La propagación de los programas compilados a través de la red se produce mediante el comando netcat 'nc'

```
#./trin.sh | nc 128.aaa.167.217 1524 &
```

```
trin.sh:
    echo "rcp 192.168.0.1:leaf /usr/sbin/rpc.listen"
    echo "echo rcp is done moving binary"

    echo "chmod +x /usr/sbin/rpc.listen"

    echo "echo launching trinoo"
    echo "/usr/sbin/rpc.listen"

    echo "echo \* \* \* \* \* /usr/sbin/rpc.listen > cron"
    echo "crontab cron"

    echo "echo launched"
  echo "exit"
```

FIG. 2-10: Esquema de instalación del TRIN00.

El paradigma de tres capas utilizado en el TRINOO (ver figura 2-11) permite que con una infraestructura mínima y sencilla se llegue a controlar un poderoso conglomerado de ordenadores conectados a Internet.

La comunicación entre las diferentes capas se realiza mediante conexiones TCP (fiables) para atacante/master, y conexiones UDP (no fiables) para *master*/*slave* y *slave*/*master* a puertos específicos de cada máquina (ver figura 2-12).

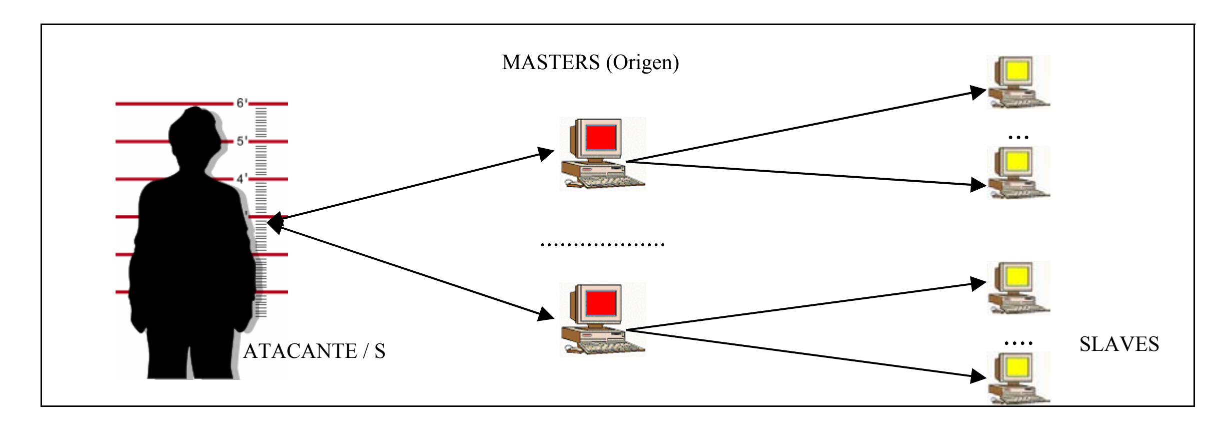


FIG. 2-11: Esquema de TRINOO.

La comunicación siempre se inicia con el envío del password. Esto permite que ni el administrador del equipo ni otros atacantes puedan acceder a controlar la red TRINOO.

	ATACANTE	MASTER	SLAVE
ATACANTE		27665/TCP	
MASTER			27444/UDP
SLAVE		31335/UDP	

FIG. 2-12: Esquema de comunicaciones entre las capas de TRINOO.

Los *daemons* situados en las máquinas *master* y *slave* son programas muy sofisticados que permiten ejecutar una serie de comandos para iniciar, controlar y parar los ataques distribuidos (DDOS). Para acceder a ellos se debe realizar una conexión TELNET al puerto especificado.

Los comandos aceptados por los *daemons* del TRIN00 son:

Master

die	Apagar el servicio.
quit	Finalizar la conexión al servicio.
mtimer *seconds*	Indica el tiempo de ataque DDOS en segundos.
dos *IP*	Iniciar ataque DDOS sobre la dirección IP especificada.

mdie *passwdord*	Parar broadcasts si el password especificado es correcto.
mping	Realizar un ping a cada máquina "activa".
mdos *ip1:ip2:ip3*	Realizar un ataque DDOS múltiple sobre las direcciones IP.
info	Visualizar información de la versión y opciones.
msize	Indicar el tamaño del buffer de los paquetes utilizados en DDOS.
nslookup *host*	Realizar una resolución de nombres(DNS).
killdead	Elimina los esclavos no activos de la lista.
usebackup	Carga la lista de esclavos activos creada por 'killdead'
bcast	Visualizar todos los ordenadores activos en la lista.
help *comando*	Proporciona ayuda sobre los comandos disponibles
mstop	Finalizar un ataque DDOS (en algunas versiones NO funciona).

Slave

aaa *password IP*	Realizar DDOS enviando paquetes UDP a la dirección IP.
bbb *password N*	Indica el tiempo máximo en segundos del ataque DDOS.
shi *password*	Envía la cadena "*HELLO*" a la lista de masters.
png *password*	Envía la cadena "PONG" al master que envió el comando.
d1e *password*	Finalizar el deminio (daemon).
rsz *size*	Especificar el tamaño del los paquetes enviados en el DOS.
xyz *password 123:ip1:ip2:ip3*	Realizar DDOS sobre las direcciones IP.

2.4.2 TRIBE FLOOD NETWORK / TFN

El TFN [Dit99-3] es otro ejemplo de herramienta que permite realizar ataques de denegación de servicio distribuido (DDOS) mediante el uso de técnicas simples de denegación de servicio (*ICMP Flood*, *SYN Flood, UDP Flood* y *SMURF*).

Además también permite obtener bajo demanda un "*shell de root*" mediante la instalación de un *daemon* que escucha en el puerto TCP requerido, lo que en la realidad permite al atacante tener un acceso ilimitado a la máquina cuando desee.

Inicialmente también se encontró en sistemas Solaris, se pensó durante un tiempo que simplemente permitía un acceso privilegiado al ordenador como puerta trasera (*back door*) para la colocación de "*sniffers*" de red.

La arquitectura de funcionamiento del TFN es bastante parecida a la del TRINOO, diferenciando entre la parte "*client*" o cliente (*masters* en TRINOO) y la parte de "*daemons*" o demonios (*slaves* en TRINOO).

Análogamente un atacante controla uno o mas clientes que a su vez controlan uno o mas demonios siguiendo el típico esquema jerárquico (ver figura 2-11).

El control de la red TFN se consigue mediante la ejecución directa de comandos (*command line*), conexión a clientes/servidores basados en UDP o ICMP o conexiones directas vía TELNET.

La comunicación entre los clientes y los demonios se realiza mediante paquetes ICMP_ECHO_REPLY, y a diferencia de TRINOO no existe la comunicación basada en TCP o UDP.

La comunicación de los comandos se realiza mediante un número binario de 16 bits en el campo ID del paquete ICMP_ECHO_REPLY. El número de secuencia es una constante 0x0000 que le hace parecer la primera respuesta a una petición PING. Los valores definidos por defecto son:

```
#define ID_ACK        123   /* for replies to the client */
#define ID_SHELL      456   /* to bind a rootshell, optional */
#define ID_PSIZE      789   /* to change size of udp/icmp packets */
#define ID_SWITCH     234   /* to switch spoofing mode */
#define ID_STOPIT     567   /* to stop flooding */
#define ID_SENDUDP    890   /* to udp flood */
#define ID_SENDSYN    345   /* to syn flood */
#define ID_SYNPORT    678   /* to set port */
#define ID_ICMP       901   /* to icmp flood */
#define ID_SMURF      666   /* haps! haps! */
```

El motivo de este cambio en la comunicación viene dado en que muchos sistemas de monitorización/protección de redes en la actualidad (Firewalls...) filtran los paquetes TCP/UDP, y especialmente aquellos que van a puertos no standares o bien conocidos (como WWW que es puerto 80, SSH que es el puerto 22...).

Sin embargo, la mayoría de sistemas dejan pasar los paquetes ICMP de ECHO y REPLY utilizados en el PING que permiten comprobar si un ordenador está encendido o no (entre otras posibles opciones).

Además, pocas herramientas de filtrado de red muestran adecuadamente los paquetes ICMP, lo que permite camuflarse entre el tráfico normal de la red.

A diferencia que en TRINOO, el cliente de TFN no está protegido por ningún password de acceso, lo que permite que sea ejecutado sin restricciones por cualquiera. El cliente TFN admite los siguientes parámetros:

./tfn *<lista_ip>* *<tipo>* [*ip*] [*port*]

<lista_ip> Contiene la lista de direcciones IP a atacar.
<tipo> -1 Tipo de máscara (spoofmask type)
0 para stop/status.
1 Para realizar UDP Flooding.
2 Para realizar SYN Flooding.
3 Para realizar ICMP Flooding.

-2 Tamaño de los paquetes a enviar.
4 Realizar un "root shell" (se debe espicifar en que puerto)
5 Realizar un ataque SUMRF, la primera IP es la dirección de origen y las demás son usadas como direcciones de broadcast.

[*ip*] Dirección de origen (separadas por @ si hay mas de una)
[*port*] Debe especificarse para SYN Flood (0 =aleatorio).

2.4.3 STACHELDRAHT

STACHELDRAHT (del alemán alambre de espinas) es una herramienta de ataques DDOS basada en código del TFN [Dit99-2] que añade algunas características más sofisticadas como el cifrado en el intercambio de mensajes.

Como en TRINOO, la arquitectura básica de STACHELDRAHT mantiene una jerarquía dónde existen los master (denominados ahora "*handlers*", manipuladores o controladores) y demonios/*daemons* o *bcast* (denominados ahora "*agents*" o agentes).

Los ataques permitidos en STACHELDRAHT al igual que en el TFN son *ICMP Flood*, *UDP Flood*, *SYN Flood* y *SMURF*. Sin embargo a diferencia del TFN no contiene la posibilidad de proporcionar un "*shell de root*" en las máquinas infectadas.

Uno de los problemas mas comunes de TFN era que las conexiones eran en claro permitiendo desde su detección por sistemas de *sniffers* hasta el robo de sesiones, lo que podía implicar la pérdida de control sobre una parte de la red controlada (pudiendo incluso llegar a perderla totalmente si todos los nodos master son víctimas de robos de conexiones).

De esta forma, el uso de técnicas de cifrado dificulta la detección de los ordenadores contaminados si examinamos el tráfico de la red y además evita el posible ataque a las conexiones de los controladotes o *handlers*.

Se implementa un servidor tipo TELNET como en TFN pero que mantiene todas las conexiones cifradas mediante un simple algoritmo de clave simétrica. El cliente acepta un único argumento que es la dirección del *handler* al que se tiene que conectar posteriormente usando el puerto 16660/TCP.

Una vez conectado se pide un password de acceso que será cifrado mediante el algoritmo BLOWFISH [WWW53] utilizando siempre la clave "*authentication*".

Como todos los programas de este tipo, STACHELDRAHT también tiene la capacidad de modificar su nombre y argumentos para no ser detectado y aparecer como un proceso corriente más del sistema.

A diferencia de las herramientas TRINOO (que usaba UDP para las comunicaciones entre *master/slave*) o TFN (que utiliza ICMP como vehículo de comunicación entre *clients/daemons*), STACHELDRAHT utiliza para sus comunicaciones entre *handlers* y agentes los protocolos ICMP y TCP indistintamente.

Otra característica que lo diferencia de TRINOO y TFT es la capacidad de actualizar a los agentes bajo demanda, es decir, de forma automática los agentes pueden pedir que se les actualice la versión de los ficheros ejecutables.

Para este fin utiliza el comando "**rcp**" que actúa en el puerto 514/TCP. Primero se bajan los nuevos binarios, luego borran los antiguos ejecutables y finalmente ejecutan los nuevos con la señal de sistema NOHUP.

La detección de sistemas comprometidos pasa por observar el tráfico de puerto 514/TCP (rcp), y examinar la dirección 3.3.3.3 que suele utilizar como dirección falsa esta utilidad cuando se produce un ataque.

2.4.4 SHAFT

Otra herramienta ampliamente conocida en los ataques de denegación de servicio distribuido y derivada de las anteriores es SHAFT [DDL00].

Como todas las herramientas distribuidas utiliza un paradigma jerárquico que se basa en la existencia de varios *masters/handlers* denominados "*shaftmasters*" que gobiernan a su vez varios *slaves/agents* que pasan a denominarse "*shaftnodes*".

El atacante se conecta mediante un programa cliente a los *shaftmasters* desde dónde inicia, controla y finaliza los ataques DDOS. SHAFT es posterior a TFN y como este utiliza el protocolo UDP para el envío de mensajes entre los *shaftmasters* y *shaftnodes*.

El atacante se conecta vía TELNET a un *shaftmaster* utilizando una conexión fiable, una vez conectado se le pide un password para autorizar su acceso al sistema. La comunicación entre los *shaftmasters* y *shaftnodes* se realiza mediante el protocolo UDP que no es fiable, por eso SHAFT utiliza la técnica de los "*tickets*" para mantener el orden de la comunicación y poder asignar a cada paquete un orden de secuencia.

La combinación del password y el ticket son utilizadas para el envío de ordenes a los *shafnodes*, que verifican que sean correctos antes de aceptarlos.

- Los *shaftnodes* entienden las siguientes órdenes:

size <*tamaño*>	Permite especificar el tamaño de los paquetes del DDOS.
type <*tipo_ataque*>	Especifica el tipo de ataque a realizar: 0 UDP Flooding 1 TCP Flooding 2 UDP/TCP/ICMP 3 ICMP
time <*segundos*>	Especifica la duración del ataque en segundos.
own <*direccion_ip*>	Añade la dirección IP a la lista de máquinas a atacar.
end <*direccion_ip*>	Elimina la dirección IP de la lista de máquinas a atacar.
stat	Visualizar las estadísticas de paquetes.
alive	Comprueba que shaftnodes del shaftmaster están activos.
switch <*h*> <*p*>	Cambia la conexión al handler/shaftmaster en el puerto p.
pktres <*pwd*><*sock*><*tickt*><*ps*>	Visualiza los paquetes enviados desde shaftnodes.

- Los *shaftmasters* entienden las siguientes órdenes:

mdos <*host*>	Inicia un DDOS al ordenador especificado (envía "own host" a todos los shaftnodes).

edos <*host*>	Finaliza un DDOS al ordenador especificado (envía "end host" a todos los shaftnodes).
time <*segundos*>	Especifica la duración en segundos del ataque.
size <*tamaño*>	Especifica el tamaño de los paquetes (máximo 8Kbytes).
type <*tipo_ataque*>	Especifica el tipo de ataque a realizar: UDP especifica UDP Flooding TCP especifica TCP Flooding ICMP especifica ICMP Flooding BOTH especifica UCP y TCP
+node <*ip*>	Añade un nuevo shaftnode a la lista.
-node <*ip*>	Elimina un shaftnode de la lista.
ns <*host*>	Realiza una petición de resolución de nombre (DNS).
lnod	Listar todos los shaftnodes.
pkstat	Estadísticas de los paquetes enviados.
stat	Visualizar status global.
switch	Asignarse como shaftmaster de los shaftnodes.
ver	Visualizar versión

Todo y no utilizar técnicas de cifrado complejas SHAFT utiliza bastante el cifrado de CESAR [RH91][Rif95][Sua99] consistente en substituir unos caracteres por otros.

	ATACANTE	SHAFTMASTER	SHAFTNODE
ATACANTE		20432/TCP	
SHAFTMASTER			18753/UDP
SHAFTNODE		20433/UDP	

FIG. 2-13: Esquema de comunicaciones entre las capas de SHAFT.

La posibilidad de cambiar los números de los puertos dinámicamente (*on the fly*) hace que SHAFT sea difícilmente detectable por sistemas convencionales (ver figura 2-13).

Al igual que los anteriores programas trata de esconderse cambiando su nombre de proceso y argumentos de llamada. Por defecto intenta hacerse pasar por un servidor WWW cambiando su nombre por el de 'HTTP'.

La conexión vía TELNET al *shaftmaster* es en claro (no usa cifrado), lo que permite descubrir el password de acceso y hacerse con el control de la red.

Por otro lado en lo que se supone un fallo de implementación, tenemos que el número de secuencia de todos los paquetes TCP es fijo siempre (0x28374839) lo que permite su detección a nivel de red.

2.4.5 TRIBE FLOOD NETWORK 2000 / TFN2K

El TFN2K [BT00] es la revisión de la herramienta TFN que permite lanzar ataques de denegación de servicio distribuido contra cualquier máquina conectada a Internet.

La arquitectura básica dónde existe un atacante que utiliza clientes para gobernar los distintos demonios instalados en las máquinas captadas se mantiene de forma que el control de este tipo de ataques mantiene la premisa de tener el máximo número de ordenadores segmentados. Así si un cliente (o master en TRINOO) es neutralizado el resto de la red continua bajo control del atacante.

Las nuevas características añadidas a TFN2K son principalmente:

- Las comunicaciones *master/slave* se realizan vía protocolos TCP, UDP, ICMP o los tres a la vez de forma aleatoria.

- Los ataques utilizados son *TCP SYN Flood*, *UDP Flood*, *ICMP/PING Flood* o *SMURF*. El demonio (*daemon*) puede ser programado para que alterne entre estos cuatro tipos de ataque, lo que permite mantener un ataque sostenido contra un ordenador concreto dificultando la detección del ataque por los sistemas tradicionales de seguridad (Firewall).

- Las cabeceras de los paquetes de comunicación *master/slave* son aleatorias, excepto en el caso del ICMP dónde siempre se utiliza *ICMP_ECHO_REPLY*, de esta forma se evita que puedan ser detectados por patrones de comportamiento.

- A diferencia de otras herramientas de DDOS, los *daemons* no responden a cada comando recibido (*acknowledge*). En su lugar, el cliente envía el mismo comando 20 veces esperando que al menos llegue una de sus peticiones.

- Los comandos enviados ya no se basan en cadenas de caracteres (*strings*), sino que son de la forma <ID> + <DATA>

<ID>	Un byte que indica (codifica) el comando a efectuar.
<DATA>	Indica los parámetros del comando.

- Todos los comandos están cifrados usando CAST-256 [RFC2612]. La clave se define en tiempo de compilación y se usa como password para acceder al cliente.

  Además todos los datos cifrados se pasan a BASE64 antes de ser enviados para asegurar que todo está en caracteres ASCII imprimibles (argucia común en prácticamente todos los ordenadores).

- Los paquetes UDP y TCP no incluyen en la pseudo-cabecera (*header*) del paquete en el cálculo del *checksum* correctamente, lo que hace que todos sean incorrectos.

- El *daemon* genera un proceso hijo para cada ataque a una dirección IP. Además prueba de diferenciarse entre sí por los argumentos/parámetros (arg[0]) que se pasan al ejecutarse.

  También altera su nombre de proceso (no su número de proceso o PID!) y los argumentos que recibe con el objetivo de pasar por un proceso común más del sistema.

Debido a estas características, es muy difícil detectar la presencia de ordenadores comprometidos en una red debido a que la comunicación es unidireccional (los *daemons* no responden a los comandos), se utilizan diferentes protocolos aleatoriamente (ICMP, UDP, TCP) y los paquetes van cifrados (CAST-256).

No obstante existe un patrón de detección que se basa en que para obtener diferentes longitudes de paquete (y dificultar su detección por tener siempre la misma longitud de paquete) TFN2K añade de uno a dieciséis '0' al final del paquete.

Al pasar a BASE64 se traslada esta secuencia al 0x41 hexadecimal ('A') con lo que la presencia de varias 'A' al final de paquetes repetidamente puede dar un indicio (NO una seguridad) de que existen ordenadores comprometidos en nuestra red.

2.4.6 Distributed reflection DOS

El enfoque clásico utilizado en las herramientas de DDOS se basa en conseguir que un gran número de máquinas comprometidas (*slaves*) dirijan un ataque directo hacía un destino concreto.

El objetivo principal es el de concentrar el mayor número de máquinas bajo nuestro control puesto que mas máquinas significará mas potencia de ataque y por lo tanto mas consumo de ancho de banda de la víctima.

Recordamos que si bien nuestro objetivo es dejar fuera de funcionamiento los servidores atacados, si degradamos mucho la calidad del servicio de comunicaciones de red (aunque los servidores sigan funcionando) el ataque puede considerarse exitoso.

Este tipo de modelo tiene varios inconvenientes, entre ellos que necesitamos obtener el control de todas las máquinas implicadas en el proceso (*masters/slaves*). Esto implica la obtención de un ataque exitoso para cada máquina bajo nuestro control y además que nuestra presencia no sea detectada por el administrador de red.

Otra opción consiste en usar técnicas de reflexión [Lar03][MP03][MP03-1]. La táctica simplemente consiste en enviar cientos de miles de peticiones a servidores con grandes conexiones a Internet (ISP por ejemplo) falseando la dirección de origen con el objetivo de que las respuestas inunden a la víctima.

Con este tipo de táctica, el atacante consigue un doble objetivo:

1. Escondemos el origen real de nuestro ataque. Esto significa que el hecho de que realicemos un ataque no pone de manifiesto inmediatamente las fuentes de este, lo que le permite seguir manteniendo un control de las máquinas. Cuando los servidores que reciben la reflexión nos corten el acceso podemos dirigirnos a otros servidores (hay cientos de grandes ISP en Internet) y continuar el ataque.

2. Ahogamos a la víctima durante un período de tiempo mayor. Si vamos cambiando los ordenadores a los cuales realizamos las peticiones, el bombardeo de peticiones continúa desde diferentes puntos de Internet, evitando cualquier posibilidad de defensa.

2.5 Defensas contra DOS/DDOS

Una vez analizados los distintos ataques de denegación de servicio DOS/DDOS realizaremos una breve introducción a los sistemas anti-DDOS. En un ataque de denegación de servicio podemos diferenciar claramente tres entidades (ver figura 2-14):

1. **Sistemas de origen:** Hace referencia a los múltiples puntos dónde se origina la generación masiva de datagramas IP.

2. **Sistemas intermedios**: Son los sistemas pasivos por dónde circulan los paquetes IP hasta llegar a su destino.

3. **Sistema final:** Destino final del ataque.

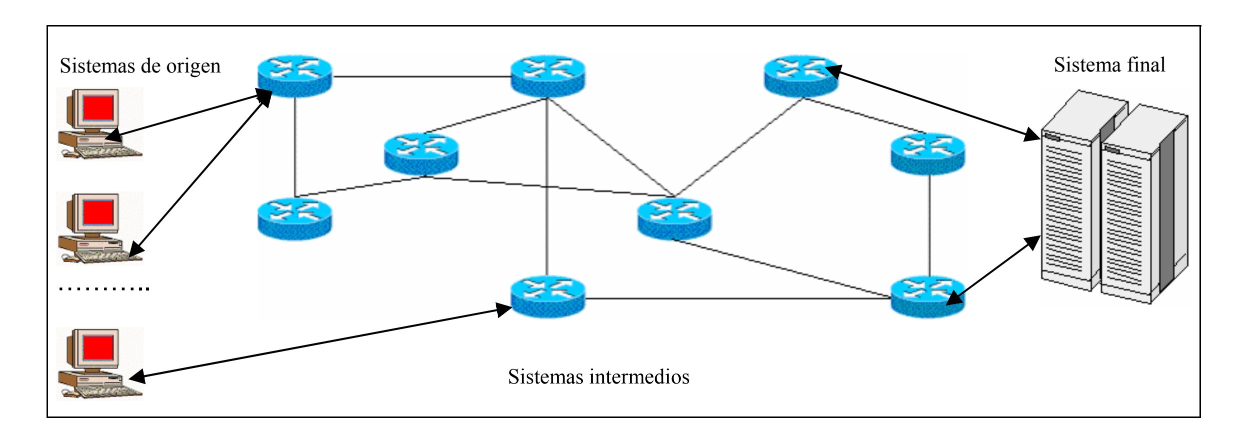


FIG. 2-14: Esquema de ataque DOS/DDOS.

Si analizamos el problema de la denegación de servicio en profundidad veremos que no hay recetas o soluciones mágicas que nos permitan evitar este tipo de ataques. Los cinco requisitos básicos que debería cumplir la solución son [MP03-2]:

1. Debemos utilizar una solución distribuida para solventar un problema distribuido. Las soluciones locales carecen de sentido puesto que no alcanzarán nunca la visión global que un ataque distribuido genera.

2. La solución propuesta no debe en ningún caso penalizar el tráfico de los usuarios legítimos. Muchas propuestas se basan en complejos sistemas de informes y filtros que mantienen listas y cuentas de todos los paquetes recibidos. Estos sistemas penalizan al global de los usuarios ralentizando el sistema.

3. Esta solución debe de ser robusta y universal. Debe ser válida para amenazas externas e internas. El sistema debe identificar[ℜ] los nodos y usuarios legítimos así como detectar y aislar si fuera necesario los nodos comprometidos o maliciosos.

4. El sistema propuesto debe de ser viable en su aplicación. La medida debe de ser adoptada por toda Internet, lo que implica que debe de ser sencilla y realizable con un coste razonable de recursos.

[ℜ] Por ejemplo mediante el uso de herramientas criptográficas.

5. Debe de ser una solución incremental. Internet es un sistema heterogéneo dónde es imposible forzar la aplicación de un nuevo protocolo. El sistema propuesto debe permitir su aplicación gradual y ser compatible con el resto de los sistemas dónde aún no esté implementada.

La mayoría de las soluciones existentes en la actualidad están basadas en sistemas clásicos de defensa como los firewalls y sistemas de detección de Intrusos (IDS). Estos se encargan de filtrar todo el tráfico existente en búsqueda de indicios de actividad maliciosa (para más información ver el capítulo de IDS).

Multops [GP01], Reverse Firewall [WWW154] o D-WARD [Obr01] son soluciones de este tipo que se encuentran instalados al final de la cadena de ataque (sistema atacado).

De esta forma, su eficacia es mínima debido a que si bien si que pueden detectar y bloquear la entrada de ataques DDOS a nuestra red, el consumo de ancho de banda se realiza igualmente, con lo que la respuesta a usuarios legítimos se ve interrumpida.

En la actualidad empiezan a surgir soluciones agregadas o distribuidas como el ACC (*Aggregated-based Congestion Control*) [Mah02], SOS (*Secure Overlay Service*) [KMR02] o DefCOM (*Defensive Cooperative Overlay Mash*) [MP03-2].

Sin embargo estas soluciones requieren de comunicación directa con el resto de sistemas intermedios, lo que significa que también deben estar instalados en el resto de los sistemas intermedios para realizar un trabajo cooperativo.

En la realidad, esta necesidad limita su eficacia y expansión únicamente a las redes de investigación, ya que al no dar una solución global ni estar aceptados como standards, casi ningún sistema existente en producción los implementa.

2.6 Ejemplo de ataque DDOS

En este último apartado del capítulo reflejaremos un ataque real de DDOS sufrido en un famoso WWW de seguridad denominado GRC.COM [Gib02].

La justificación del porqué este ataque en concreto es el examinado en este capítulo viene dada por dos factores:

1. Pese a que los ataques de denegación de servicio son muy abundantes y hay gran constancia de ellos [WWW10][WWW11][WWW55][WWW56][WWW57], la información disponible sobre estos ataques suele ser mínima, reduciéndose a pequeñas notas en organismos como el CERT [WWW21] o boletines electrónicos como SECURITYFOCUS [WWW24] y grandes titulares de prensa o televisión con mas contenido peliculero que realidad técnica. En este caso concreto, existe un detallado análisis disponible de forma pública que disecciona perfectamente el ataque recibido.

2. Como se ha podido comprobar, los ataques DDOS suelen ser una generalización de los ataques DOS, con lo que un ejemplo de ataque distribuido es suficientemente complejo como para obtener una idea de cómo funcionan realmente.

El 11 de enero de 2002 a las 02:00 am (*US Pacific time*) el servidor WWW.GRC.COM fue atacado mediante una inundación masiva de paquetes de petición de conexión (*TCP SYN Flood*) utilizando la técnica **DRDOS** (*Distributed Reflection Denial Of Service*).

De repente, las dos conexiones T1 (1.5Mbits/s) mostraron un tráfico de salida nulo, ya que el servidor WWW era incapaz de completar una petición de conexión y servir contenidos. Sin embargo, el tráfico registrado en las conexiones de red era de casi el 100% de su capacidad 2 x T1 (ver figura 2-15).

Analizando los paquetes recibidos de los ISP QWEST, Verio y Above.net no se observa ninguna anomalía y parecen paquetes TCP SYN / ACK legítimas hacia el puerto 80 de GRC.COM dando como puerto de origen de la petición el 179.

Cabe destacar que todas las peticiones provienen de ISP (*Internet Service provider*), es decir, organizaciones con conexiones a Internet de un gran ancho de banda, lo que permite una gran capacidad de generación de tráfico.

Dirección IP	**Nombre (DNS)**
129.250. 28. 1	Ge-6-2-0.r03.sttlwa01.us.bb.verio.net
129.250. 28. 3	ge-1-0-0.a07.sttlwa01.us.ra.verio.net
129.250. 28. 20	ge-0-1-0.a12.sttlwa01.us.ra.verio.net
205.171. 31. 1	iah-core-01.inet.qwest.net
205.171. 31. 2	iah-core-02.inet.qwest.net
205.171. 31. 5	iah-core-01.inet.qwest.net
206. 79. 9. 2	globalcrossing-px.exodus.net
206. 79. 9.114	exds-wlhm.gblx.net
206. 79. 9.210	telefonica-px.exodus.net
208.184.232. 13	core1-atl4-oc48-2.atl2.above.net
208.184.232. 17	core2-atl4-oc48.atl2.above.net
208.184.232. 21	core1-atl4-oc48.atl2.above.net
208.184.232. 25	core2-core1-oc48.atl2.above.net

FIG. 2-15: Análisis real de los paquetes 179/TCP SYN recibidos.

El puerto 179 es el destinado al servicio BGP (*Border Gateway Protocol*) que permite que diferentes sistemas autónomos se comuniquen e intercambien sus tablas de rutado (*routing tables*) para conocer que redes/ordenadores tienen accesibles en cada momento.

Muchos sistemas troncales conectados a Internet admiten conexiones al puerto 179/TCP para el intercambio de tablas de rutado con otros sistemas mediante el protocolo BGP.

En la figura 2-16 podemos observar cómo se produce un ataque de DDOS indirecto utilizando un servicio existente de forma correcta pero falseando la dirección de origen:

1. Se crea la infraestructura necesaria para el DDOS (instalación de los ordenadores master/slaves).

2. Inicio del ataque DDOS consistente en enviar desde los esclavos miles de peticiones "correctas" de conexión TCP al puerto 179 a diferentes ISP. Se falsea la dirección de origen de la petición y el puerto, colocando la dirección IP de la víctima.

3. Obtención de cientos de miles de respuestas de los distintos ISP que bloquearán todo el ancho de banda del servidor WWW atacado.

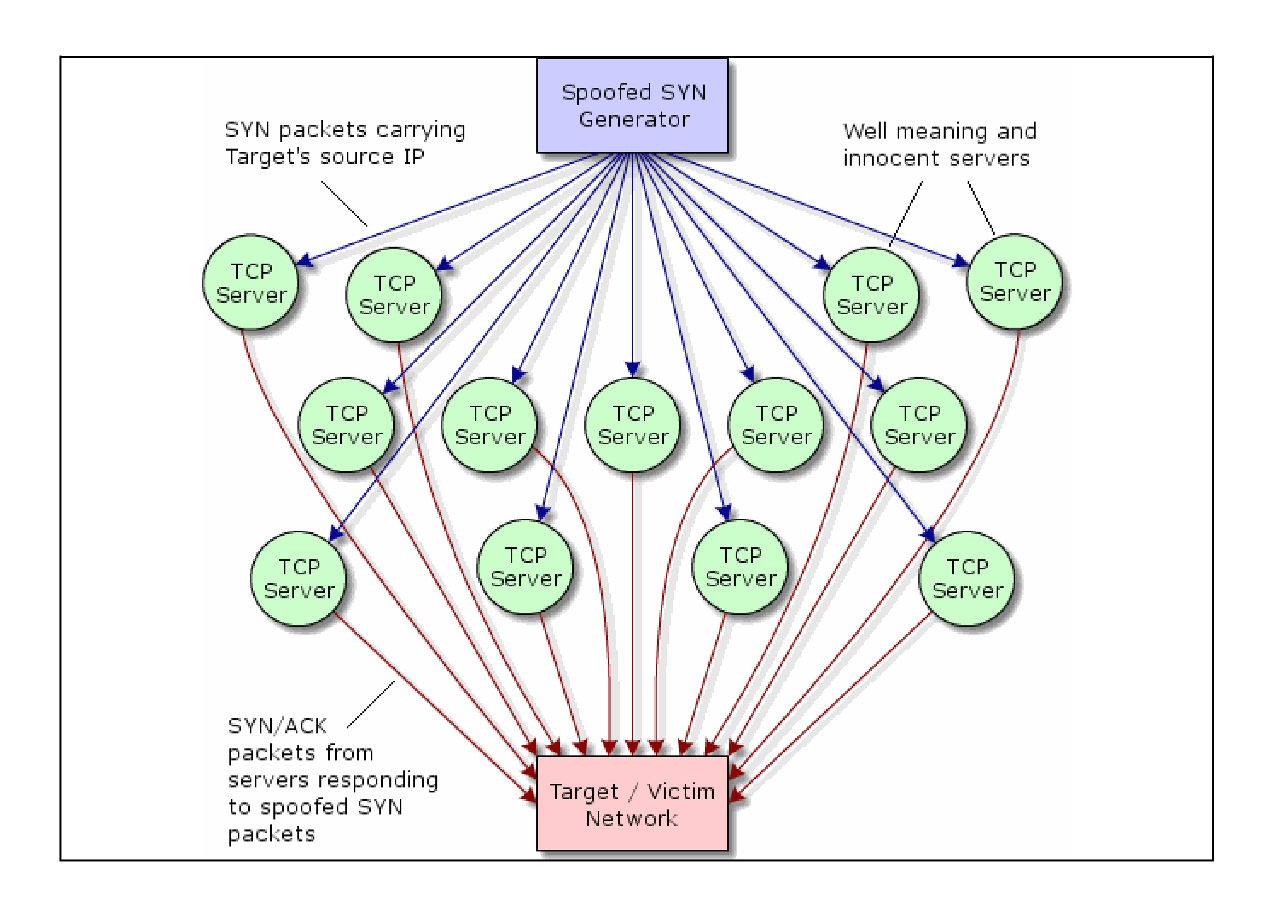


FIG. 2-16: Esquema del ataque DRDOS.

Una vez conocida la arquitectura del ataque, comenta Steve Gibson que tardó mas de tres horas en conseguir que su propio ISP (con servicio 24x7) bloqueara todos los paquetes de conexiones que provenían del puerto 179/TCP.

Una vez conseguido el bloqueo del trafico proveniente del puerto 179/TCP, el servidor WWW continuó sin estar operativo debido a otros ataques de inundación de paquetes

(*TCP/UDP Flood*) a los puertos 22/TCP (SSH), 23/TCP (TELNET), 53/UDP (DNS), 80/TCP (WWW), 4001/TCP (Proxy), 6668/UDP (IRC)[ℜ].

La explicación dada a que inmediatamente después de bloquear el primer ataque continuara atacado el WWW es que este segundo ataque se lanzó simultáneamente con el primero, sin embargo, debido a que el primero proviene de ISPs con una gran capacidad de ancho de banda, habían enmascarado el segundo.

A continuación en la figura 2-17 se pueden observar algunas muestras tomadas de las peticiones de *TCP SYN Flood* obtenidas del segundo ataque. Se puede observar que muchas de ellas provienen de ordenadores conectados a redes que tienen grandes conexiones a Internet (.nasa.gov o .yahoo.com)

Dirección IP	**Nombre (DNS)**
64.152.4.80	www.wwfsuperstars.com
128.121.223.161	veriowebsites.com
131.103.248.119	www.cc.rapidsite.net
164.109. 18.251	whalenstoddard.com
171. 64. 14.238	www4.Stanford.edu
205.205.134.1	shell1.novalinktech.net
206.222.179.216	forsale.txic.net
208. 47.125. 33	gary7.nsa.gov
216. 34. 13.245	channelserver.namezero.com
216.111.239.132	www.jeah.net
216.115.102. 75	w3.snv.yahoo.com
216.115.102. 76	w4.snv.yahoo.com
216.115.102. 77	w5.snv.yahoo.com
216.115.102. 78	w6.snv.yahoo.com
216.115.102. 79	w7.snv.yahoo.com
216.115.102. 80	w8.snv.yahoo.com
216.115.102. 82	w10.snv.yahoo.com

FIG. 2-17: Análisis real de los paquetes del segundo ataque.

Después de aplicar los filtros correspondientes, el ISP de GRC.COM (Verio) descartó un total de **1.072.519.399** paquetes maliciosos de *TCP SYN Flood*.

[ℜ] En [WWW58] se puede obtener una lista completa de los servicios asociados a los puertos TCP y UDP.

2.7 RESUMEN

En este capítulo hemos comentado la evolución histórica de los ataques sobre redes en Internet y su evolución final a los ataques de denegación de servicio. Se han comentado profundamente los modelos de funcionamiento de los ataques DOS y DDOS así como los fundamentos de los protocolos básicos de Internet que posibilitan este tipo de ataques (IP, TCP e ICMP).

En la primera parte del capítulo dedicada a la denegación de servicio (DOS), se han analizado los principales ataques documentados (IP Flooding, Broadcast, Smurf...) así como su impacto en las redes de ordenadores.

En la segunda parte se explican los motivos de la evolución de los ataques de denegación de servicio a ataques distribuidos (DDOS) así como las principales herramientas existentes que permiten la realización de estos ataques (TFN, TFN2K...).

Posteriormente se realiza una explicación sobre los posibles métodos anti-DOS/DDOS (D-WALL, Reverse Firewall, SOS) y las dificultades e implicaciones que conlleva la aplicación de estas medidas.

Finalmente se describe un ataque de DDOS real documentado por GRC.COM dónde se analizan los pormenores de la técnica "*Reflexion-DDOS*" empleada por el atacante para mantener el WWW fuera de servicio durante varias horas.

CAPITULO 3

IDS

En este tercer capítulo realizaremos un estudio sobre los sistemas de detección de intrusos o IDS (*Intrusion Detection System*). Diferenciaremos entre los distintos tipos de sistemas existentes y nos centraremos en los sistemas de detección de intrusos para redes de ordenadores o NIDS (*Network IDS*).

Se analizarán los diferentes componentes, arquitecturas y configuraciones que forman los sistemas NIDS. También se comentarán los diferentes protocolos y paradigmas standard que existen en la actualidad para la comunicación entre los distintos componentes de sistemas IDS.

Describiremos las distintas técnicas de análisis utilizadas sobre datos obtenidos por los distintos componentes del IDS así como los efectos derivados de la detección automática de intrusos en redes de ordenadores. Se introducirán los conceptos de falsos positivos y falsos negativos.

También se enumerarán los principales inconvenientes y limitaciones que presenta la utilización de sistemas IDS/NIDS así como las futuras líneas que pueden ir marcando la evolución de los NIDS, centrándonos en los sistemas de prevención o IPS (*Intrusion Prevention System*).

Finalmente comentaremos un ataque típico y clásico en la bibliografía de los sistemas de detección, el ataque del famoso hacker Kevin Mitnick.

3.1 Firewalls

Durante mucho tiempo el mecanismo de seguridad en redes más extendido ha sido únicamente el uso de un *firewall*. Este sistema nos permite de una manera simple y eficaz aplicar filtros tanto para el tráfico de entrada como para el de salida en nuestra red.

Podemos diferenciar entre dos políticas básicas de configuración de firewalls [Nor99]:

- **Permisividad máxima** (*allow everything*) dónde el uso de filtros es mínimo o inexistente. Esta política permite prácticamente la circulación de todo el tráfico y se utiliza principalmente en Intranets/LAN, campus universitarios y organizaciones dónde la libertad de uso de aplicaciones (o la gran cantidad de ellas) es necesaria para el funcionamiento ordinario del sistema.

  Es una política que dificulta enormemente el uso de otros sistemas y deja a la red muy vulnerable a prácticamente cualquier tipo de ataque interno o externo.

En estos casos se recomienda segmentar la red en dominios y acotar cada uno de estos dominios, ya que raramente todos los ordenadores tienen que acceder a todos los recursos disponibles de la red.

- **Permisividad mínima** (*deny everything*) aplica la política contraria a la anterior. En este caso se deniega acceso a todos los servicios de la red y se van permitiendo accesos a estos a medida que se necesiten.

  De esta forma es bastante improbable que recibamos un ataque a un servicio que desconocíamos que teníamos en la red. Por otro lado, el trabajo de otros sistemas se facilita enormemente ya que pueden configurarse para que detecten fácilmente cualquier comportamiento anómalo en la red (simplemente se debe monitorizar los accesos a los servicios y comprobar si esos accesos están permitido expresamente o no).

  Cabe notar que este tipo de política requiere un gran esfuerzo ya que es poco flexible y en organizaciones con gran cantidad de usuarios con diferentes requerimientos puede llevar a tener que permitir tantos accesos cruzados que deje de ser práctico.

Destacar que el simple uso de un firewall puede crear una falsa sensación de seguridad que de nada sirve si no son configurados y "mantenidos al día" (aplicación de los *parches*/*patches* del fabricante, supervisión y adaptación al tráfico de la red...).

Muchas organizaciones con cientos de ordenadores y decenas de firewalls no disponen de una sola persona cualificada asignada exclusivamente a su mantenimiento!!

Por otro lado, este tipo de sistemas son incapaces de detectar tipos de ataques más sofisticados (DOS por ejemplo), lo que hace necesario la adopción de otros sistemas de control para completar la seguridad en nuestra red.

3.2 Historia de los IDS

La historia sobre los sistemas de detección de intrusos en ordenadores empieza en 1972 cuando James P. Anderson de las fuerzas aéreas norteamericanas (USAF) publica un texto sobre la seguridad en los ordenadores [Bru01].

Este tema empieza a cobrar fuerza paralelamente al desarrollo de la informática, puesto que cada vez hay mas procesos "críticos" controlados por ordenadores y los militares temen cualquier cosa que no controlan.

Algunos estudios se realizan durante los años siguientes hasta que en 1980 James P. Anderson escribe "**Computer Security Threat Monitoring and Surveillance"** [And80], dónde se inician las bases de la detección de intrusos en sistemas de computadores principalmente mediante la consultas de ficheros de log.

Entre 1984 y 1996, Dening y Neummann desarrollan el primer modelo de IDS denominado IDES (Intrusion *Detection Expert System*) basado en reglas. A partir de este momento, se han ido proponiendo y creando nuevos sistemas de detección de intrusos [MC01] hasta obtener una separación clara entre los sistemas que efectúan la detección dentro de los ordenadores y aquellos que la efectúan en el tráfico que circula por la red.

3.3 IDS

Podemos definir la detección de intrusos (*Intrusion Detection* o ID) como *"un modelo de seguridad aplicable tanto a ordenadores como a redes. Un sistema **IDS** recolecta y analiza información procedente de distintas áreas de un ordenador o red de ordenadores con el objetivo de identificar posibles fallos de seguridad. Este análisis en busca de intrusiones incluye tanto los posibles ataques externos (desde fuera de nuestra organización) como los internos (debidos a un uso abusivo o fraudulento de los recursos).*

Los sistemas IDS suelen utilizar técnicas de análisis de vulnerabilidades (a veces referenciado en bibliografía como scanning), es decir examinan todos nuestros sistemas en búsqueda de alguna vulnerabilidad." [WWW59]

Un sistema de detección de intrusos o **IDS** (*Intrusion Detection System*) es un paradigma que por su naturaleza intrínseca de supervisión de los recursos es aplicado tanto a ordenadores como a redes de computadores [And80]:

- En el caso de los ordenadores se realiza a nivel de sistema operativo para controlar los accesos de los usuarios, modificación de ficheros del sistema, uso de recursos (CPU, memoria, red, disco...) con el objetivo de detectar cualquier comportamiento anómalo que pueda ser indicativo de un abuso del sistema.

  En la bibliografía [Fran02] a veces pueden encontrarse como **HIDS** (*Host Intrusion Detection System*).

  Un ejemplo de aplicación de este tipo de herramientas en sistemas operativos de ordenadores puede ser el conocido software TRIPWIRE. [WWW60]

- En el caso de redes de ordenadores pueden monitorizarse usos de anchos de banda, accesos a/desde direcciones no permitidas, uso de direcciones falsas... con el objetivo de encontrar un comportamiento anómalo o atípico en el tráfico de la red supervisada que nos pueda indicar una posible anomalía.

  También pueden referenciarse en la bibliografía como **NIDS** (*Network Intrusion Detection System*) [Gra00][HFC02][WWW94].

En este capítulo nos centraremos en los sistemas de detección de intrusos (IDS) aplicados a redes de ordenadores (NIDS).

Todo y que el concepto de IDS engloba el de NIDS, no toda la bibliografía acepta esta segunda diferenciación dentro de los sistemas de detección de intrusos, con lo que nosotros utilizaremos IDS o NIDS indistintamente,

3.3.1 Monitorización de redes en tiempo real

Realizar la supervisión de una red de comunicaciones implica necesariamente realizar un estudio detallado y pormenorizado de todo el tráfico que circula por esta. Si hacemos unos simples cálculos podemos observar que resulta del todo inviable (por no escribir imposible) filtrar toda la información que circula por nuestra red en tiempo real.

Para demostrar esta afirmación, realizaremos un sencillo pero gráfico estudio sobre los estándares de las redes mas populares actualmente (tanto LAN como WAN) y el tráfico que generan en un día completo[ℜ].

Para poder realizar este cálculo de forma completa, necesitamos también conocer el tamaño de los datagramas IP que circularán por la red. Como este tamaño es imposible de calcular debido a que un datagrama IP tiene un tamaño máximo de 64K, realizamos una estimación real a partir del tráfico de Internet. Según el estudio de Kc Claffy y Sean McCreary [CMC00] podemos asumir un tamaño medio típico para los datagramas IP que circulan por Internet es de 1500 Bytes (ver figura 3-1).

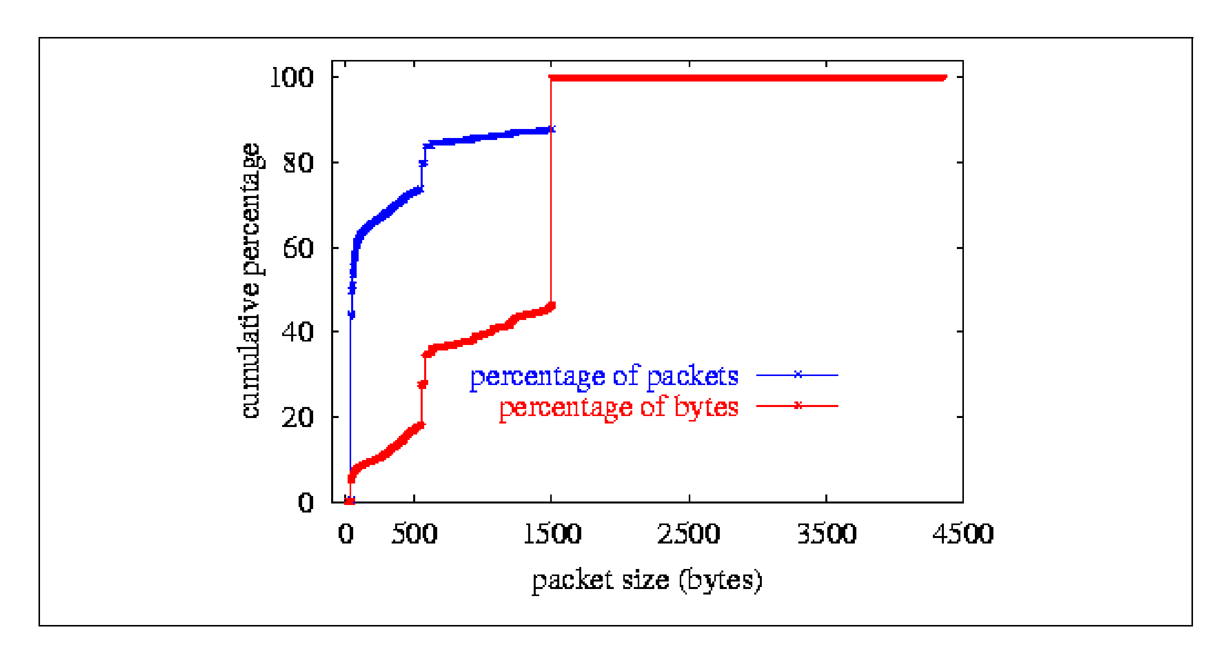


FIG 3-1: Porcentaje acumulativo de datagramas IP en INTERNET en el año 2000.

[ℜ] 1 Día = 24 Horas = 24 * 60 minutos = 24 * 60 * 60 segundos = **86.400** Segundos.

En este análisis consideraremos que todo el tráfico generado en estas redes es tráfico IP a monitorizar, cosa que no es cierta puesto que los datagramas IP van encapsulados en *frames* (capa 2 del modelo OSI) que dependen de la tecnología que use la red.

Sin embargo, esta asunción nos dará una cota superior de la capacidad necesaria para procesar todos los paquetes IP, ya que también asumimos como inexistente el tiempo de análisis de cada paquete (nuestro objetivo es saber que potencia necesitamos para el proceso en tiempo real).

De esta forma la asunción de que todo el tráfico es IP queda nivelada con la asunción de que el tiempo de proceso de cada paquete que circule por la red es cero (ver la figura 3-2).

Tecnología **T**	Espacio de disco Cantidad de información transmitida por día. EDD = (T * 86400 s) / 1 GByte	Potencia de procesador Número de paquetes por día. **PP** = EDD / 1500 Bytes
ADSL (256Kbits/s)	(32768 Bytes/s * 86400 s) / 1Gbyte = **2,6 Gbytes**	**1.887.436**
ADSL (2Mbit/s)	**21,09 Gbytes**	**15.099.494**
LAN (10 Mbit/s)	**105,4 Gbytes**	**75.497.472**
LAN (100 Mbit/s)	**1054,6 Gbytes**	**754.974.720**
WAN ATM (155 Mbit/s)	**1634,7 Gbytes**	**1.170.210.816**
LAN (1 Gbit/s)	**10546,8 Gbytes**	**7.549.747.200**

FIG. 3-2: Anchos de banda y tamaño diario de las redes más comunes.

Por otro lado, los procesadores más potentes en el momento de realizar este trabajo funcionan a 3GHz. Con lo que necesitaríamos procesar cada paquete **en un ciclo de reloj** (cosa totalmente imposible) ya que necesitaríamos un ciclo de reloj para recibir el paquete, uno para procesarlo y otro para retransmitirlo si fuera necesario y no quisiéramos eliminarlo.

Los ordenadores actuales acceden a memoria externa y disco cuyas velocidades distan mucho de los 3GHz del procesador. Por otro lado, con las tecnologías superiores al Gigabit (que ya existen actualmente) como la fibra OC48 (2.48Gbits/s), esto queda aún mas lejos.

Como se ha demostrado anteriormente, el filtrado de todo el tráfico generado por una red es imposible en tiempo real (tanto en potencia de proceso CPU como en espacio de almacenamiento), lo que implica que en caso de realizarse este proceso debería ser a posteriori (*off-line*) con lo que su utilidad baja mucho.

Si bien es cierto que es muy importante conocer que se ha recibido un intento de ataque, conocerlo días o semanas después no sirve como argumento ante los consejos de dirección o superiores.

Cabe notar que el almacenamiento de esta información es **básico** para realizar análisis forense (*computer forense*), sin embargo nuestro objetivo (y por lo tanto el de este estudio) no debe ser el de examinar los registros y logs cuando nuestro sistema ya ha sido comprometido, seducido y abandonado por el hacker de turno.

3.3.2 Signatures (Firmas)

Sin embargo, en cualquiera de los paquetes que circulan por la red puede encontrarse un intento de ataque, un ataque en toda regla o incluso el paseo triunfal de un hacker que ya ha conseguido entrar en algún sistema. ¿Qué podemos hacer?

Definiremos una **firma** (*signature*) como *"aquello que define o describe un patrón de interés en el tráfico de nuestra red."* [Nor99]

Análogamente, diremos que un **filtro** (*filter*) *"es la trascripción de una firma (signature) a un lenguaje compresible por los sensores que monitorizan nuestra red."* [Nor99]

Las firmas (*signatures*) nos permiten diferenciar entre todo el tráfico generado por la redes y obtener un subconjunto de este lo suficientemente pequeño como para que sea tratable computacionalmente y lo suficientemente amplio como para poder detectar comportamientos anómalos en tiempo real (o casi).

Las firmas utilizadas en IDS usualmente son simples patrones que permiten detectar ataques ya conocidos (*Smurf*, *Land*...). Su funcionamiento es el mismo que el de los antivirus, se basan en encontrar coincidencias de ataques ya conocidos en el tráfico actual de la red.

Algunos productos IDS permiten la creación de filtros por el usuario (ver figura 3-3), lo que facilita la adaptación del sistema a nuestras necesidades. Sin embargo, realizar filtros útiles necesita al menos de:

- Que el administrador de seguridad esté cualificado (conozca perfectamente el protocolo IP y su propia red).

- Que el IDS proporcione un lenguaje suficientemente potente como para poder expresar nuestras necesidades.

- Que los filtros (tanto los creados por defecto como los personalizados) sean revisados/modificados/ampliados frecuentemente.

```
Filter pptp ip()
  {
      # El ataque "Land" se basa en dar como dirección de origen y destino la misma IP

       if (ip.src == ip.dest)
          {

            # Guardamos la hora y direcciones MAC e IP de origen y destino

            record system.time, eth.src, ip.src, eth.dst, ip.dst to land_recdr;
          }
  }
```

FIG 3-3: Ejemplo de un filtro para el ataque *Land*.

La capacidad de crear filtros permite una gran flexibilidad y potencia en la detección de tráfico anómalo. Un ejemplo de esto podrían ser los filtros que se realizan sobre un protocolo en concreto (por ejemplo en HTTP) y que permiten incluso guardar log de las conexiones que envían/reciben determinada información (como por ejemplo los filtros contra pornografía).

Hay que tener en cuenta que este tipo de filtros pueden atentar contra la intimidad de las personas y que deben ser conformes a la ley vigente (LORTAD en España).

Obviamente, como pasa en los antivirus, la actualización de las firmas debe ser constante ya que usualmente un nuevo tipo de ataque no será detectado por el sistema IDS que simplemente se dedique a buscar patrones en el tráfico de la red.

El uso de un firewall bien configurado (nos limitará la cantidad de tráfico a procesar) así como la personalización adecuada de los filtros aplicados en la red, nos dotarán de un sistema seguro y útil en el que podemos confiar [WWW66].

3.3.3 Eventos de interés (EOI)

Una vez demostrado que no es práctico realizar el filtrado de todo el tráfico producido en una red, debemos centrarnos en ir refinando los resultados obtenidos para conseguir un conjunto razonablemente pequeño de muestras que nos permita el proceso en tiempo real.

Definiremos los **eventos de interés** (*Events Of Onterest, EOI*) como el subconjunto mínimo de muestras que debemos analizar para considerar nuestra red "segura" o como el subconjunto de los genuinos positivos verdaderos (*genuine non-false positives*) [Nor99].

Este subconjunto se obtiene a partir del resultado de aplicar los filtros, firmas y reglas personalizadas del sistema de detección de intrusos al tráfico total (ver figura 3-4).

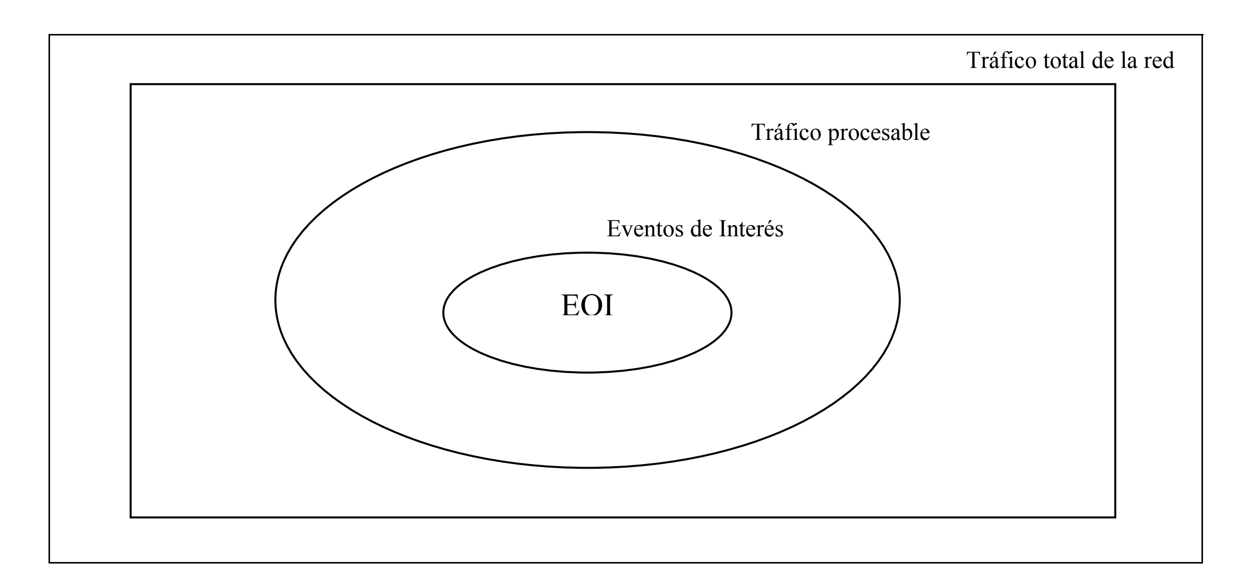


FIG 3-4: Eventos de interés (EOI).

Muchos estudios proponen una fórmula de evaluación de riesgos para los eventos de interés detectados (EOI) y que nos puede ayudar a cuantificar numéricamente la gravedad del evento detectado [Car00][Car01][Nor99] [WWW69]. Los aspectos a tener en cuenta en esta fórmula son (ver figura 3-5):

1. **Criticidad** (*Criticality*). No todos los ordenadores tienen la misma función. De esta forma un ataque a un servidor DNS, a un firewall o a un PC Cliente se valoran de distinta forma.

2. **Letalidad** (*Lethality*). Los diferentes ataques (*exploits*) no tienen siempre el mismo objetivo. Dependiendo de si simplemente tiene posibilidades de funcionar en algún sistema no parcheado a si permite bloquear la máquina (un ataque de tipo DOS por ejemplo) o ganar acceso como usuario simple o root, evaluaremos el ataque.

3. **Contramedidas** (*countermeasures*). Una vez detectado un ataque (o un inicio de posible ataque), nuestra capacidad de respuesta es otro factor a tener en cuenta y que pondera en la fórmula (de aquí la importancia de IDS en tiempo real). Las contramedidas a aplicar pueden ser de sistema (bloquear una cuenta de usuario, parar un servicio temporalmente o incluso apagar la máquina) o de red

(interceptar las comunicaciones desde una dirección determinada, ajustar anchos de banda de entrada y salida de la red...)

Severity = (Criticality + Lethality) – (System countermeasures + network countermeasures)

FIG 3-5: Fórmula de evaluación de la gravedad de un EOI.

De esta forma, de todos los eventos de interés detectados por son sensores podemos obtener una representación numérica en la consola que nos facilite de una forma rápida un orden de prioridad.

3.4 Arquitecturas de los NIDS

Según la aproximación utilizada para la monitorización del tráfico de la red, en la bibliografía [Tim01] se agrupa los NIDS en tres grupos distintos:

a) **Signature based NIDS**: Al igual que muchos antivirus, estos IDS se basan en la búsqueda de patrones conocidos (firmas) en el tráfico de la red.

b) **Anomaly based NIDS**: Se basan en analizar el tráfico de la red creando estadísticas y asignándoles pesos. Cuando se detecta tráficos sospechoso, se confronta con las estadísticas anteriores y en función del peso asignado y la cantidad de ocurrencias del evento se dispara una alarma o no.

c) **Protocol modeling NIDS**: Estos sistemas de detección de intrusos buscan paquetes que contengan anomalías o configuraciones poco comunes de los protocolos de red (a fin de cuentas, los datos van encapsulados en distintos datagramas de distintos protocolos).

Existen dos componentes básicos para cualquier sistema de detección de intrusos (IDS) que son los sensores y la consola (ver figura 3-6):

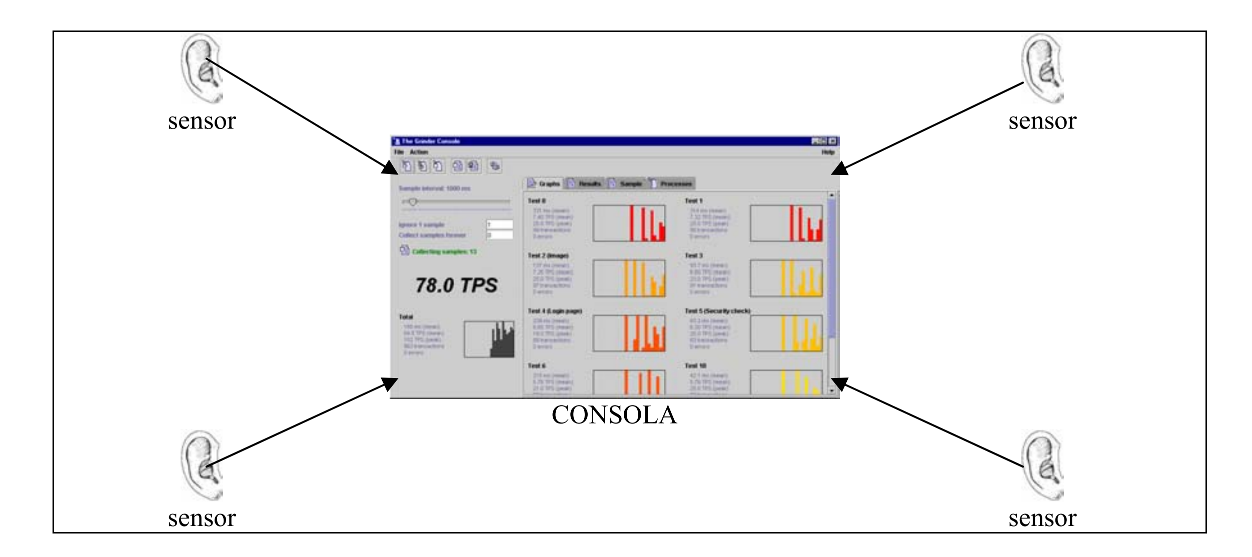


FIG 3-6: Elementos básicos de un IDS.

1. Los **sensores** (*sensors*) de un IDS son elementos pasivos que examinan todo el tráfico de su segmento de red en búsqueda de eventos de interés. Dependiendo del paradigma que utilicen para comunicarse con la consola del sistema detector de intrusos pueden clasificarse en dos grupos [Nor99][WWW72][WWW73]:

    push: Cuando se detecta un evento de interés el sensor crea un paquete de datos que envía a la consola. Un protocolo muy utilizado con este tipo de sensores es el SNMP que permite la definición de *traps* para el envío de información a un receptor (la consola en este caso).

    Su principal inconveniente es que pueden ser observador por el atacante para descubrir cómo ha sido configurado y ante que tipo de patrones reacciona, lo que permite establecer que patrones ignora el sensor y por tanto cuales emplear en un ataque.

pull: Almacenan los eventos de interés hasta que la consola pregunta por ellos al sensor. Si se establece un protocolo de intercambio de mensajes cifrado y se pregunta regularmente a los sensores puede ofrecer un mecanismo eficaz de comunicación con la consola.

Para evitar susceptibilidades, en caso de que el sensor no detecte eventos incluirá una cadena de caracteres aleatoria para evitar enviar siempre el mismo paquete de cuando no existen eventos detectados.

2. La **consola** (*console*) de un sistema de detección de intrusos se encarga de recibir toda la información de los sensores (ya sea mediante "*pull*" o "*push*") y presentarla de forma entendedora al operador. Desde la consola se pueden configurar los distintos sensores así como emprender acciones en respuesta a los datos recibidos de los sensores.

3.4.1 CIDF

El **CIDF** (*Common Intrusion Detection Framework*) [WWW79] es un proyecto iniciado a finales de la década de los noventa por la oficina de información y tecnología (*Information technology Office*) del DARPA [WWW80].

Inicialmente se enmarcó como parte de programa de vigilancia de la información de redes de ordenadores, pero progresivamente, conforme los sistemas de detección de intrusos han alcanzado suficiente importancia, se convirtió en una rama independiente apoyada por otros organismos internacionales como el grupo IDWG (*Intrussion Detection Working Group / Exchange Format*) de IETF [WWW81].

Su objetivo es el de crear interfaces de aplicaciones (API) y protocolos que permitan la comunicación entre los diferentes IDS con el objetivo de reaprovecharlos para otros sistemas [Lee00][Ruo00][Wan00].

CIDF (ver figura 3-7) nos propone una serie de estructuras que denomina cajas (*boxes*) que encarnan cada una de las distintas funciones que deberían realizar los sistemas IDS.

De esta forma, simplemente define las funcionalidades genéricas deseadas y sus interconexiones, dejando abierto a cada fabricante la implementación final de estas. Los componentes principales son:

- **Event generator** *(E boxes):* Estas cajas describen la función de los sensores del sistema de detección de intrusos. Su función es recoger información de la red y generar informes/alertas para la consola de monitorización.

- **Analysis** *(A boxes):* Son las encargadas de procesar la información obtenida de los sensores y realizar su análisis. Se admite como ampliación de su funcionalidad [Nor99] la capacidad de realizar recomendaciones al operador e incluso de actuar proactivamente.

- **Database** *(D boxes):* Esta entidad simboliza la base de datos (o base de conocimiento) dónde se almacenan los informes, firmas y patrones detectados en la red por el IDS. El objetivo de mantener esta información es doble:

    1. Obtener la posibilidad de generar informes históricos y permitir el uso de técnicas de *Business Intelligence* y *Data Warehouse* (BI/DW) para la extracción de información y obtención de nuevo conocimiento.

    2. Proactividad del sistema. Conseguir una respuesta rápida del IDS ante un ataque nuevo consultando otros ataques similares registrados.

- **Response** *(R boxes):* Este elemento es el encargado de proporcionar una respuesta ante los eventos obtenidos de las otras cajas (*E/A/D Boxes*) o sugerir acciones al operador de consola (bloquear el acceso desde una dirección IP, limitar el numero de conexiones nuevas aceptadas por segundo....).

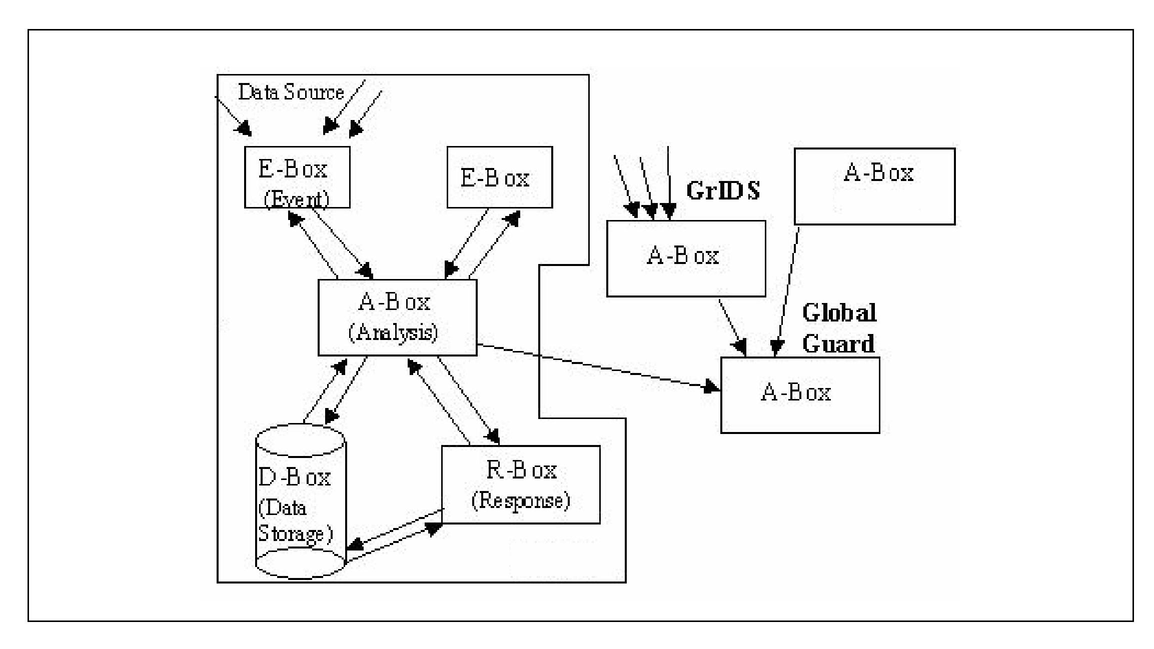


FIG 3-7: Ejemplo de uso del modelo CIDF.

Muchas veces existe la tendencia de economizar esfuerzos y recursos, instalando el sensor y la consola en la misma máquina. Esta arquitectura se desaconseja totalmente, ya que como hemos comentado en puntos anteriores, los sensores deben filtrar el tráfico (cosa que requiere de mucha potencia) y añadir la consola u otros programas (firewalls...) no hace mas que cargar el sistema y restarle potencia al sensor.

3.4.2 DIDS

En grandes organizaciones (multinacionales) o universidades (dónde hay diferentes facultades, departamentos, laboratorios...) un único sistema IDS no proporciona la flexibilidad necesaria para la heterogeneidad de los elementos de que disponemos.

Los sistemas **DIDS** (*Distributed Intrusion Detection System*) [Ein01][Vil02] proporcionan este servicio de detección de intrusos para grandes redes.

El análisis de los DIDS es tan o mas complejo que el realizado con los NIDS, con lo que queda fuera de este trabajo aunque se cita la bibliografía correspondiente.

Su característica diferenciadora respecto a los sistemas NIDS tradiciones, es la presencia de dos elementos nuevos en su arquitectura (ver figura 3-8):

- **Central Analysis Server**: Es el centro del sistema DIDS y es el encargado de recibir toda la información procedente de los agentes y realizar un repositorio común de conocimiento. También realiza las funciones de control y sincronización de los diferentes nodos que forman parte del sistema.

- **Co-operative Agent network**: Es un sistema autónomo encargado de la monitorización de una red. Detecta posibles incidentes e informa al servidor central para que comunique a todos los nodos el ataque detectado así como las contra-medidas a realizar. Dependiendo de la implementación, el agente puede llegar a tomar contra-medidas de forma autónoma (auque siempre informando y supeditándose al servidor central).

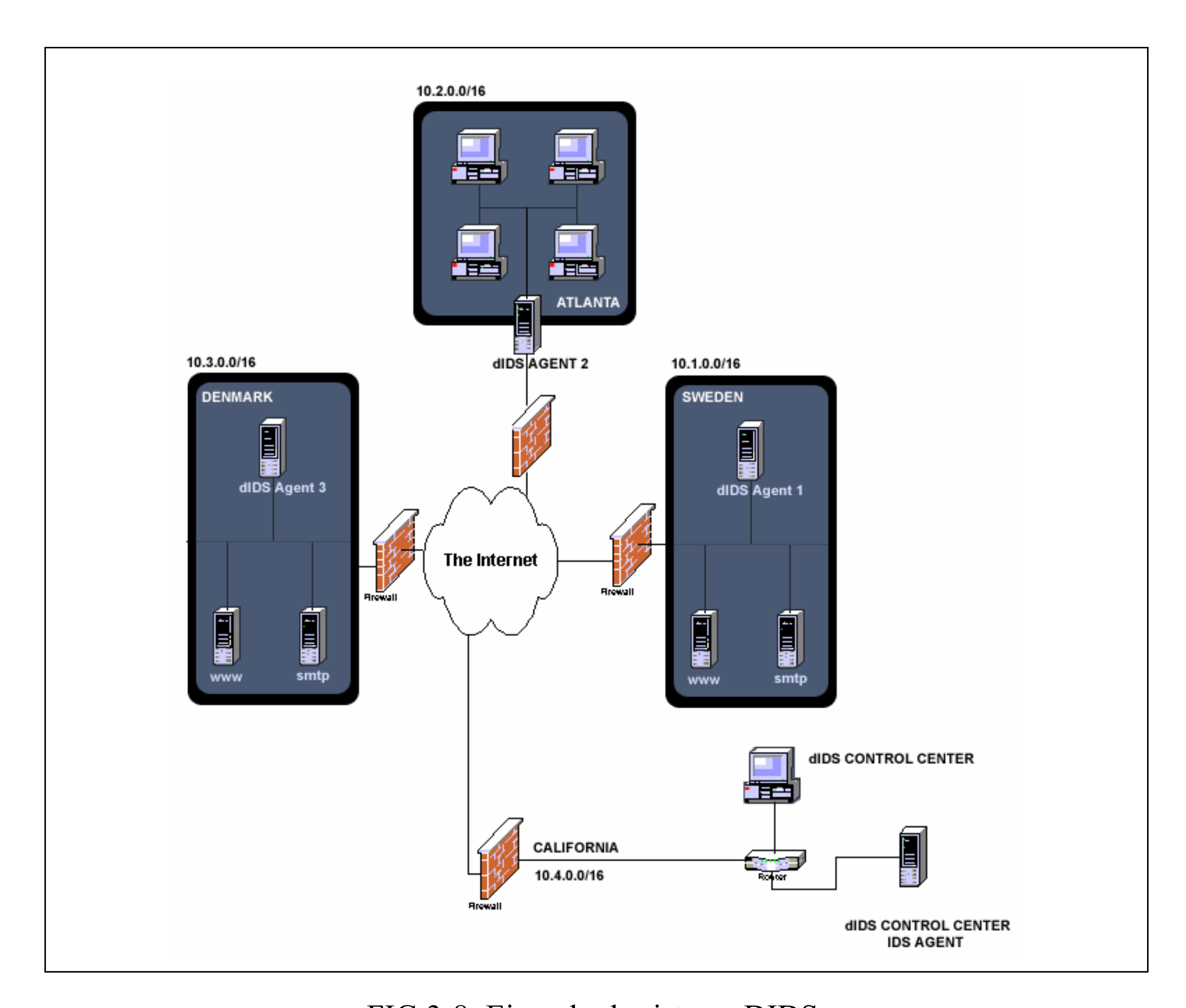


FIG 3-8: Ejemplo de sistema DIDS.

3.5 Ubicación de los NIDS

Una vez explicados los componentes básicos de un sistema IDS, se ha de decidir de qué tipo han de ser los distintos sensores que utilizará el NIDS (*pull* o *push*), y finalmente se ha de concretar en que lugar o lugares de la red deben colocarse:

A) **Antes del firewall** (ver figura 3-9): Esta arquitectura se basa en detectar todos los paquetes que llegan a nuestra red antes de ser filtradas por el firewall. De esta forma, realizamos una búsqueda de eventos de interés en todo el tráfico recibido sin interferencias de filtros del firewall.

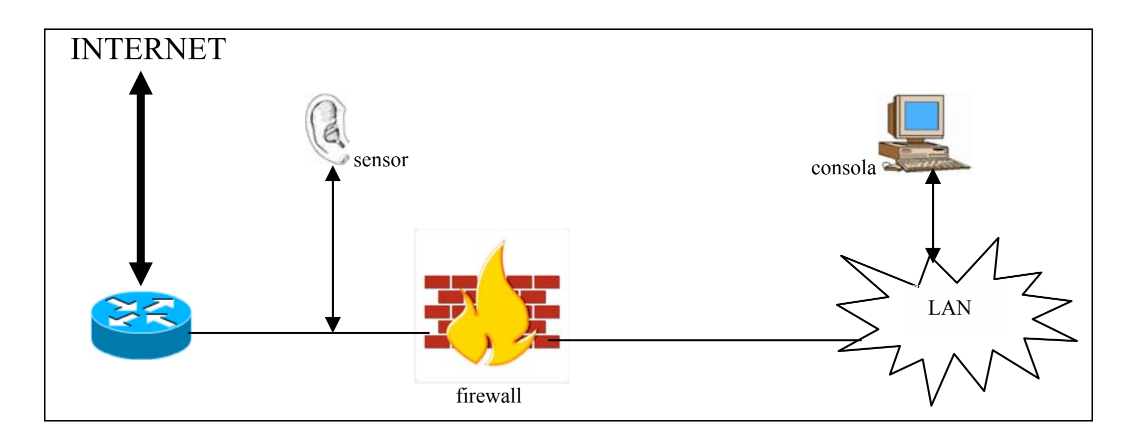


FIG 3-9: Sensores antes del firewall.

B) **En el firewall o adyacente** (ver figura 3-10): Consiste en situar el sensor en el propio firewall. De esta forma se evitan ataques de intrusos a los sensores externos y se eliminan muchos falsos positivos, ya que procesamos únicamente el tráfico que el firewall deja pasar.

C) **Antes del firewall y en el firewall o adyacente** (ver figura 3-11): Es la opción es más costosa pero que ofrece mayor seguridad, ya que permite obtener lecturas del tráfico total y del tráfico filtrado por el firewall. Permite examinar la configuración del firewall y comprobar si filtra correctamente o no.

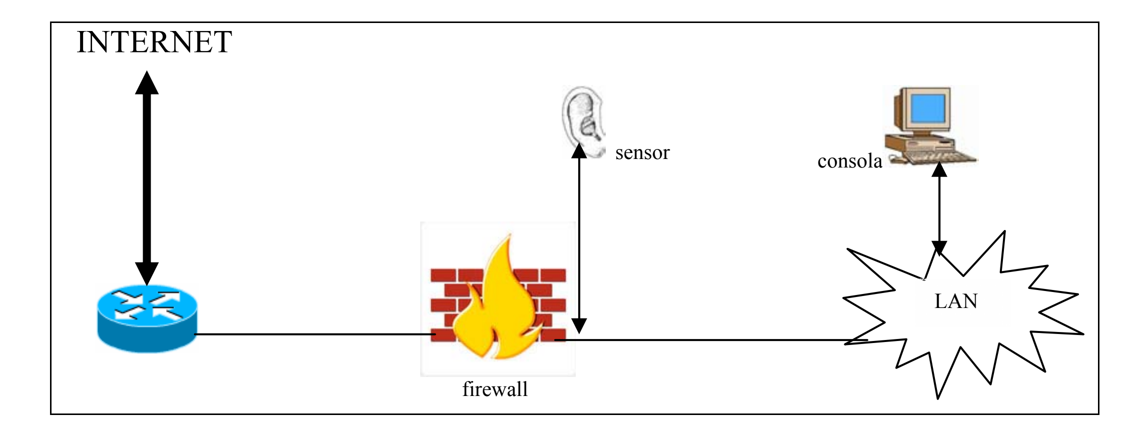


FIG 3-10: Sensores en el firewall.

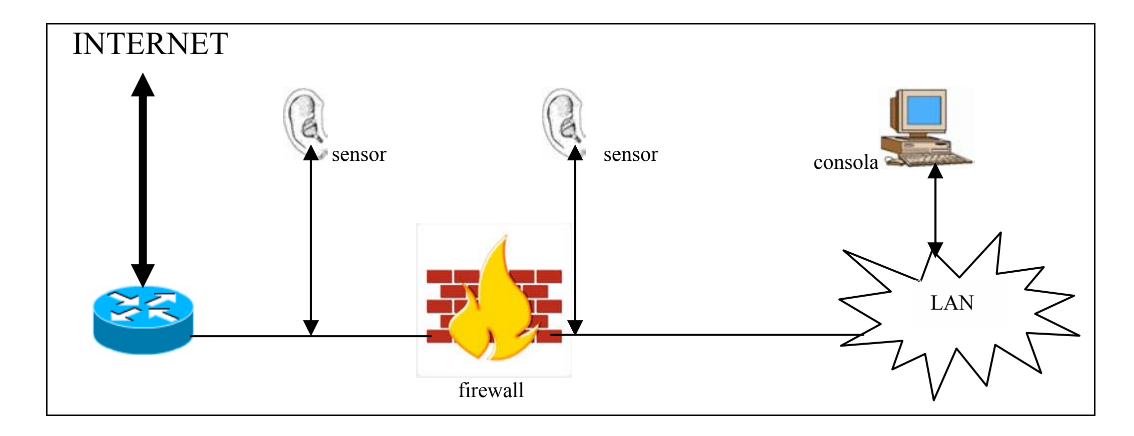


FIG 3-11: Sensores antes y en el firewall.

3.6 Protocolos de comunicación Sensor-Consola

Como siempre suele pasar en informática, cuando una compañía desarrolla un nuevo producto suele acompañarlo de un protocolo propietario encargado de gobernar las comunicaciones entre las diferentes partes del sistema. En los primeros productos (o primeras versiones) de sistemas de detección de intrusos esto ha sido una constante, lo que obligaba al usuario a depender de un único fabricante/producto.

Conforme se han ido extendiendo los productos IDS y el mercado ha ido creciendo, los usuarios y expertos demandaban la búsqueda de un lenguaje/protocolo común, ya que si simplemente quiero compartir mis datos con mi propio ISP para mejorar la seguridad o

detectar nuevas firmas de ataques desconocidos, estábamos obligados ambos a poseer el mismo producto (a veces incluso era necesaria la misma versión!!) o esto es del todo imposible.

Además, muchas veces los ataques a redes de ordenadores se concentran en determinadas máquinas que sostienen los servicios vitales de la empresa o universidad (servidores de ficheros o impresoras, servidores de nombres,...). Esto es debido a que muchos atacantes se nutren de listas de ordenadores denominadas *shopping lists* (listas de la compra) que contienen las direcciones IP de los servicios ofrecidos en cada red.

Estas listas circulan por Internet en servidores WWW *undergrounds* y listas de discusiones de temática hacker.

Poder compartir los datos obtenidos por nuestro IDS con el de otras organizaciones u empresas puede evitar problemas en un futuro, ya que usualmente los atacantes utilizan las mismas técnicas en las diferentes organizaciones (obtenidas usualmente de listas de la compra) con la esperanza de encontrar una que sea vulnerable al ataque perpetrado.

Si los diferentes IDS tienen un lenguaje común, la publicación de nuevas firmas/trazas de ataques detectados será más rápida, de forma que el primero que lo detecte puede compartir sus nuevos filtros/firmas con los demás que podrán utilizarlas directamente en su IDS sin tener que adaptarlas.

Para conseguir esta interrelación entre distintos productos IDS se han propuesto diversos protocolos, a continuación enumeramos los más importantes:

CISL (*Common Intrusion Specification Language*) [Nor99][WWW79]

Junto con la especificación de CIDF, DARPA e IETF [WWW80][WWW81] propusieron un standard de representación de la información asociada a sistemas de detección de intrusos basado en un lenguaje de expresiones regulares de tipo S dónde se insertan marcas (*tags*) y datos (ver figura 3-12).

Estas expresiones están delimitadas por paréntesis imitando el lenguaje LISP y permiten expresar tres tipos de información:

1. *Raw Event Information*: hace referencia a información básica obtenida directamente sin ningún tipo de post-proceso. Principalmente tráfico de red y resultados de auditorias (*audit trail).*

2. *Analysis Results*: Descripciones de anomalías o ataques detectados.

3. *Response prescriptions*: Conjunto de acciones de respuesta predefinidas para algunos comportamientos observados (ej. Bloquear acceso desde una dirección IP si detectamos más de 1.000 conexiones por segundo procedente de ella).

```
(Login
        (Location    (Time '07:07:07 09 Julio 2003')  )
        (Initiator     (Hostname 'gabriel.lsi.upc.es')   )
)
```

FIG 3-12: Ejemplos de una expresión formal en CISL.

OPSEC (*Open Platform for Secure Enterprise Connectivity*) [Nor99]

Esta propuesta parte de la empresa privada de seguridad Checkpoint Software [WWW78], que como suele ocurrir cuando una compañía lidera un sector intenta imponer su standard mediante una gran cuota de mercado.

El paradigma propuesto a finales de 1997 permite agrupar en un único modelo las distintas necesidades y capacidades de un modelo de seguridad. OPSEC se fundamenta en el uso de interfaces de aplicaciones (API) que permiten la interconexión de los distintos módulos, protocolos standards y un pseudo-lenguaje de alto nivel tipo "*shell script*" que permite gobernar todos los elementos y funciones (por ejemplo permite la interconexión con sistemas de firewalls para la adopción de contra-medidas ante comportamientos sospechosos).

OPSEC es el standard "*de facto*" para la industria en la comunicación entre sistemas IDS y Firewalls ya que ha sido adoptado por más de 250 compañías.

CCI (*Common Content Inspection*) [Nor99]

Análogamente al OPSEC, las diferentes compañías lideradas por Chekpoint [WWW82][WWW83] propusieron otro standard para el análisis de la información contenida en los paquetes que recorren la red. En lugar de las cajas (*boxes*) propone:

- *Content redirector:* Servicio que redirecciona el contenido a los motores de inspección y análisis.

- *Content Inspector:* Es el servicio que realiza la inspección y el análisis de los contenidos.

EL API propuesto está muy interrelacionado con OPSEC (nacen de la misma empresa de seguridad) y ambos se modifican teniendo en cuenta esta dependencia funcional.

ANSA (*Adaptive Network Security Alliance*) [Nor99]

La empresa de seguridad ISS también propuso su standard para la interoperatividad de distintos sistemas de seguridad [WWW84][WWW85]. Sus principales características son:

- *Automated Response*: Permite reconfigurar dinámicamente a los distintos equipos de red (firewalls, switches…) para responder a las amenazas detectadas en "tiempo real".

- *Lockdown*: Esta característica hace referencia a "encerrar" o "enjaular" las distintas direcciones IP de los atacantes para evitar que puedan acceder o reconfigurar servicios críticos de la red (DNS, servidores de mail…).

De esta forma, aunque nuestro gestor de correo sea vulnerable, si nuestro sistema detecta que la dirección es hostil proactivamente no le permitirá acceder a él.

- *Decision Support*: La funcionalidad de asistencia para la toma de decisiones permite la utilización de técnicas de data mining (correlación...) y consultas históricas a bases de datos para sugerir acciones ante las posibles amenazas detectadas.

- *Security Management*: Históricamente, los elementos de seguridad existentes en las redes informáticas se encontraban diseminados sin ningún tipo de control centralizado que los agrupara. El control/supervisión centralizado de los diferentes elementos de seguridad de nuestra red nos permitirá mantener una coherencia en nuestras actuaciones. Cada uno de estos elementos en lugar de tomar decisiones aisladamente de forma unilateral se comunican con el resto del sistema.

3.7 Análisis de los datos obtenidos por los sistemas NIDS

Una vez comentadas las distintas arquitecturas de sistemas de detección de intrusos y sus protocolos de comunicación así como sus diferentes elementos, pasamos a profundizar en el análisis de los datos obtenidos por nuestros sistemas de seguridad.

Tal como se demostró anteriormente el tamaño de disco necesario para almacenar el tráfico de un solo día es totalmente desorbitado (ver figura 3-2). Con los filtros aplicados al tráfico supervisado y los eventos de interés disminuimos substancialmente esta cantidad, sin embargo, no solamente nos interesa el tráfico registrado en un solo día, sino que nos interesan históricos de la semana, mes o incluso de años.

Toda esta cantidad de información debe ser almacenada de forma óptima para poder ser consultada ágilmente. De otra forma, dejará de ser utilizada y por lo tanto de ser útil.

El problema de almacenar/recuperar información es un tema muy amplio que no trataremos en este documento. Sin embargo sí comentaremos que últimamente muchos productos IDS empiezan a incorporar soporte de bases de datos relacionales como elementos importantes de soporte para sus sistemas de información (Oracle, SQL Server, MySQL, PostgreSQL...).

Sea cual sea el formato usado para el almacenamiento de la información debe cumplir las siguientes características:

1. Debe realizar una reducción importante del volumen de datos pero conservando la información importante.

2. Debe poseer un formato compacto, fácil de consultar, escribir y actualizar.

3. Debe permitir la interrelación (cruce) de todos los datos entre sí.

El formato **TCP QUAD** [Nor99][WWW86] es una reducción compacta de la información contenida en los paquetes IP que se basa en almacenar una cuádruple tupla que contiene los siguientes campos[ℜ]:

(Fecha, Dirección origen, Dirección de destino, Tipo de protocolo)

El objetivo de almacenar históricos de tráficos de red es doble, por un lado poder realizar informes y estadísticas sobre incidentes en nuestra red. Por otro lado nos debe permitir conocer cuando un ataque presenta características similares (o iguales) a otro recibido anteriormente, ya que podemos obtener información de cómo reaccionar ante él de forma proactiva ("en tiempo real"), evitando ser sujetos pasivos y lamentarnos posteriormente.

Definiremos la **correlación** como la relación mutua entre dos o más elementos de un conjunto dado. De esta forma, gran parte de las consultas a las bases de datos se basarán únicamente en buscar correlaciones entre los eventos de interés detectados y los datos históricos almacenados.

[ℜ] Dependiendo de la bibliografía estos campos varían ligeramente. Otros autores proponen añadir el campo de opciones (flags o banderas).

- **Correlaciones por dirección de origen**: Se basan en encontrar similitudes entre conexiones provenientes de la misma fuente (por ejemplo un escáner de puertos a nuestra red detectará que desde el mismo origen se realizan miles de peticiones a distintos puertos).

- **Correlaciones por dirección de destino**: Considera las conexiones que tienen como destino la misma dirección IP (por ejemplo un DOS. Tenemos miles de peticiones hacia un mismo destino).

- **Correlaciones de firmas** *(crafted packed signatures)*: Se basan en buscar conexiones desde/hacia un puerto determinado (muchos virus y programas de *backdoor* basan su comunicación en puertos no standards). También se buscan configuraciones no estandards de los distintos campos del paquete IP, como por ejemplo las banderas o *flags* (hay varios ataques de DOS que se basan en activar todas las opciones de los datagramas IP para ver si el sistema operativo no sabe como reaccionar ante ellos y queda inutilizado).

- **Correlaciones de contenidos**: Estas correlaciones hacen referencia al contenido (datos) de los distintos paquetes de información que circulan por la red. Buscar patrones como "*/etc/passwd*" en conexiones TELNET o FTP, inspeccionar conexiones HTTP en busca de "*EXEC C:\WINNT\COMMAND\FORMAT*"...

Otras posibilidades que nos brinda la correlación es la de poder evaluar la importancia (criticidad) de un posible incidente (ver figura 3-13).

Por ejemplo podemos observar la periodicidad de los paquetes recibidos, lo que nos permite por ejemplo saber de que ancho de banda dispone el atacante "a priori". Si es una simple prueba rutinaria de un hacker "a ver que pesca" enviando peticiones muy espaciadas en tiempo a distintos servicios y direcciones IP, o un ataque en toda regla a un servidor concreto.

El sistema IDS debe permitir realizar estas consultas interactivamente al operador de la consola para investigar libremente en las bases de datos. Además deben realizarse de forma automática sólo para los eventos de interés de extrema criticidad, ya que el

proceso de correlación es muy lento y colapsaría el sistema, de forma que cuando detectásemos el ataque ya sería demasiado tarde para actuar.

```
10:09:34.123   atacante1.com.echo > maquina_atacada.es.echo icmp echo request
10:09:34.131   atacante1.com.echo > maquina_atacada.es.echo icmp echo request

δ2 - δ1 = 10:09:34.131 - 10:09:34.123 = 8 milésimas


13:19:43.23   atacante2.com.echo > maquina_atacada1.es.echo icmp echo request
13:19:46.21   atacante2.com.echo > maquina_atacada2.es.echo icmp echo request

δ2 - δ1 = 13:19:46.21 - 13:19:43.23 = 2.37 segundos
```

FIG 3-13: Ejemplos de trazas para evaluar la criticidad.

Existe una gran controversia en la bibliografía [WWW88] sobre la cantidad de tiempo (semanas, meses, años?) que debemos tener almacenada en la base de datos del IDS para poder realizar consultas tanto el operador como el propio sistema automáticamente.

Obviamente, lo deseable sería tener todo el histórico de la red, sin embargo debemos tener en cuenta que:

- Más tiempo implica mas espacio de disco (crecimiento exponencial).
- Más tiempo implica una ralentización de las búsquedas (pérdida de eficiencia).
- Más tiempo no implica necesariamente más seguridad.
- En caso de necesitar la informática forense (*computer forense*) nuestro histórico debe permitir la reconstrucción total los actos sucedidos en la red.
- También debemos tener en cuenta las leyes vigentes de protección de datos y almacenamiento de logs (LORTAD...)

Por lo que una solución propuesta [Nor99] aboga por una ventana de al menos tres meses (90 días).

A partir de esta fecha, los datos se deben "reducir" o "compactar" (mediante TCP QUAD por ejemplo) para almacenarlos en otra base de datos de históricos. La normativa de la administración americana PDD63 establece una ventana de 72 a 96 horas dónde se deben poder realizar trabajos de reconstrucción forense.

3.8 Falsos positivos y falsos negativos

En los puntos anteriores hemos analizado las diferentes partes que integran un esquema IDS. También hemos comentado los diferentes protocolos utilizados para conseguir la comunicación entre los distintos programas existentes en el mercado así como varias herramientas que se utilizan para la detección de los posibles incidentes (firmas, reglas, correlaciones...).

Un punto básico a tratar tras el análisis de las muestras obtenidas (eventos de interés) de nuestra red es el de la detección de **falsos positivos** y **falsos negativos**.

*"Un **falso positivo** (false positive) es un término aplicado a un fallo de detección en un sistema de alertas (usualmente en sistemas antivirus o de detección de intrusos). Sucede cuando se detecta la presencia de un virus o una intrusión en el sistema que realmente no existe." [WWW87]*

*"Un **falso negativo** (false negative) es un término que hace referencia a un fallo en el sistema de alerta (usualmente en sistemas antivirus o de detección de intrusos). Sucede cuando un virus o una intrusión existe en nuestro sistema y es **'permitida'** (ignorada o no detectada) por el sistema de alerta." [WWW87]*

Los falsos positivos pueden agruparse en cinco grupos dependiendo de la naturaleza de su origen [Tim01]:

- **Reactionary Traffic alarms:** Se detecta un comportamiento sospechoso como consecuencia de tráfico generado anteriormente (generalmente no malicioso). Por ejemplo la detección de muchas respuestas "*ICMP network unrecheable"* procedentes de un router porque el equipo destino no se encuentra operativo o accesible en esos momentos.

- **Equipment-related alarms:** Las alarmas del NIDS detectan paquetes dentro del tráfico de la red que identifica como no "usuales". Esto puede ocurrir por ejemplo con balanceadores de carga, puesto que generan paquetes específicos para el control de todos los nodos.

- **Protocol Violations:** Estos avisos se producen por software mal programado (bugs) o que implementan de forma incorrecta o anticuada algunas partes de los protocolos de Internet.

- **True False Positives:** Todos aquellos falsos positivos que no se encuadren en ninguna de las categorías anteriores.

- **Non Malicious alarms:** Alarmas producidas al detectar rastros de comportamientos maliciosos pero que en ese contexto determinado no lo son. Si publicamos en nuestra página WWW el código de un virus analizándolo, cada vez que una persona descargue la página creará una alerta en el IDS porque detectará el virus en nuestra red.

Obviamente, nuestro sistema de detección de intrusos debe producir los mínimos falsos positivos posibles y **ningún** falso negativo (porque con uno sólo, ya tenemos al intruso en nuestro sistema, y toda la inversión en seguridad se vuelve inútil y de difícil justificación).

Según Kevin Timm [Tim95] el 90% de las alarmas detectadas por sistemas de detección de intrusos son falsos positivos, con lo que tan sólo el 10% son realmente peligrosas. Si personalizamos el NIDS a nuestra red, generalmente con los sistemas convencionales podemos llegar a reducir los falsos positivos a un 40% del total de alarmas.

Adhitya Chittur [Chi01] afirma en su tesis "**Model generation for an intrusion detection system using genetic algorithms**" que con modelos genéticos se puede bajar la tasa de falsos positivos hasta menos del 10%.

Por otro lado, hemos de tener en cuenta que un IDS que reporte cientos de alertas diarias dejará de ser útil y muy probablemente lleve a que alguna alerta verdadera sea ignorada por los operarios entre tantos avisos.

En la figura 3-14 podemos observar 9 peticiones de establecimiento de conexión en un segundo procedentes de "*maquina1.com*" para "*maquina2.es*". Generalmente y a priori el IDS detectaría una masiva llegada de peticiones desde una misma dirección IP en un lapso de tiempo corto: Estamos recibiendo un Ataque!

```
09:09:34.123   maquina1.com.601 > maquina2.es.login S 13961:13961(0) win 4096
09:09:34.153   maquina1.com.601 > maquina2.es.login S 13962:13962(0) win 4096
09:09:34.159   maquina1.com.601 > maquina2.es.login S 13963:13963(0) win 4096
09:09:34.183   maquina1.com.601 > maquina2.es.login S 13964:13964(0) win 4096
09:09:34.323   maquina1.com.601 > maquina2.es.login S 13965:13965(0) win 4096
09:09:34.823   maquina1.com.601 > maquina2.es.login S 13966:13966(0) win 4096
09:09:34.943   maquina1.com.601 > maquina2.es.login S 13967:13967(0) win 4096
09:09:34.980   maquina1.com.601 > maquina2.es.login S 13968:13968(0) win 4096
09:09:34.998   maquina1.com.601 > maquina2.es.login S 13969:13969(0) win 4096
```

FIG 3-14: Ejemplos de falso positivo para SYN FLOOD.

Esta conclusión que para la mayoría de casos es correcta (¿porque nuestra "maquina2.es" debe recibir tantas peticiones de conexión casi simultáneas?), no siempre lo es de forma absoluta.

Si "maquina2.es" es por ejemplo un punto de entrada hacia nuestra red corporativa puede pasar simplemente que a primera hora (09:09 horas) se conecten muchos empleados para realizar su trabajo. Sin embargo, esto no explica que todas las direcciones de origen sea la misma. ¿Por qué alguien se conectará 9 veces en un segundo?

Un ejemplo práctico y real de falso positivo puede ser el uso de **NAT** (*Network Address Translation*) [Has97][WWW90] en redes corporativas. Todas las máquinas de una misma red realizan sus conexiones al exterior desde una única dirección IP.

Esta metodología se utiliza mucho por motivos de seguridad (DMZ por ejemplo), económicos (es más barato una dirección IP que 10) o incluso técnicos (ADSL, MODEM-cable...).

De esta forma, podemos observar que una situación potencialmente peligrosa puede simplemente ser un uso normal de los recursos de red.

En la actualidad y debido a la política de todos los ISP de tecnologías domésticas de conexión a Internet (ADSL, Cable…) se están implementando "*proxys*" [WWW91] automáticos de acceso para contenido HTTP. De esta forma, las peticiones de cualquier usuario de ADSL a un servidor WWW son automáticamente redireccionadas al PROXY de su proveedor de conexión.

Los ejemplos de falsos negativos son mucho mas complicados de encontrar en bibliografía y exponer en este escrito, ya que implicaría que alguien consiguió un acceso no autorizado a una red "segura" y a nadie le hace gracia reconocer esto en "públicamente" (muchas veces no se reconoce ni en privado). Generalmente, los falsos negativos suelen producirse por:

1. **Configuración deficiente de los recursos de la red:** Tener varios elementos de seguridad (IDS, firewalls, VPN...) no es suficiente para considerar nuestra red segura. Estos deben estar convenientemente configurados y adaptados a su medio (el tráfico de las redes no es estático y varía).

2. **Ataques desde dentro**: Muchas veces se obvia un uso pernicioso de los recursos de nuestra red desde dentro. El atacante no siempre es un hacker de un país extraño que se dedica a hacer el mal a quien puede. Tener controles internos también permitirá detectar programas o troyanos encargados de facilitar acceso desde dentro a posibles atacantes.

3. **Equipos no parcheados y víctimas de los últimos "*exploits*"**: Tener servidores con versiones de software anticuadas son un reclamo excesivamente apetitoso y que ningún sistema de seguridad puede proteger.

3.9 IPS

En los últimos artículos y congresos, la bibliografía [Des03][WWW109] recoge una de las tendencias futuras de los sistemas de detección de intrusos, los sistemas de prevención de intrusos o **IPS** (*Intrusion Prevention System*).

Definiremos un *"sistema de prevención de intrusos (Intrusion Prevention System, IPS) como un dispositivo (hardware o software) que tiene la habilidad de detectar ataques tanto conocidos como desconocidos y reaccionar a esos para impedir su éxito."* [Des03]

Los sistemas de prevención de intrusos pueden verse como la evolución de dos elementos que han dominados la seguridad en todas las redes informáticas del mundo:

- **Firewall**: Es el elemento que garantiza o bloquea el acceso a los recursos de nuestra red.

- **IDS**: Permite mantener el estado de las conexiones y examinar el contenido de los paquetes que circulan por nuestra red.

Los IPS pueden agruparse en cinco categorías dependiendo de su arquitectura y ubicación dentro de la red a proteger:

1. **Inline IPS**: Estos sistemas de prevención de intrusos se caracterizan por colocarse entre la red y el ordenador (o tramo de red) a proteger. Su arquitectura se basa en dos tarjetas de red (ver figura 3-15):

- La primera conectada a la red exterior y que no dispone de dirección IP. Su función es la de realizar *bridging* transparente entre la red y el IPS. Al no disponer de dirección IP, este interface es totalmente invisible y no puede recibir ningún tipo de tráfico expreso (ni ser atacado).

- La segunda tarjeta esta conectada a la red "segura" o interna y nos permite conectarnos al sistema IPS para su gestión y configuración.

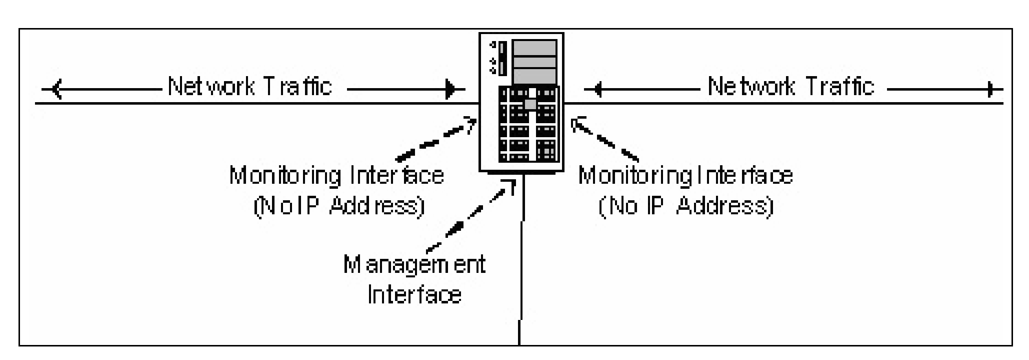


FIG 3-15: Ejemplos de IPS inline.

El sistema IPS procesa todo el tráfico entrante y saliente de manera que cada paquete puede pasar (*forward*), eliminarlo (*drop*), grabarlo en el log (*log*), borrarlo del log (*unlog*) o incluso rescribir el paquete eliminando elementos potencialmente peligrosos (*rewrite*) [HL01][Des03].

Estos sistemas deben ser muy sólidos, ya que si el IPS deja de funcionar (fallo hardware/software), se pierde todo acceso a la red.

2. **Layer seven switches**: Los conmutadores o switches son dispositivos de capa 2 del modelo OSI [Ric98-1]. En el caso de las redes IP, los diferentes protocolos para los distintos servicios (HTTP, FTP, SSH...) se sitúan en la capa 7 del modelo OSI. De esta forma, para poder inspeccionar paquetes IP de diferentes servicios necesitamos un conmutador de nivel 7.

   Estos IPS contienen un componente hardware muy importante que le proporciona una velocidad de proceso en el filtrado de paquetes de red muy superior a los sistemas convencionales basados en un programa que se ejecuta

en un ordenador. Por otro lado, la flexibilidad de reconfiguración y nuevas versiones de los IPS basados en programas queda sacrificada al ser un elemento hardware en su casi totalidad (ver figura 3-16).

Su modo de funcionamiento se basa en un único puerto del *switch* encargado de mantener la conexión de red con el exterior, y el resto de puertos conectados a los distintos servidores a monitorizar.

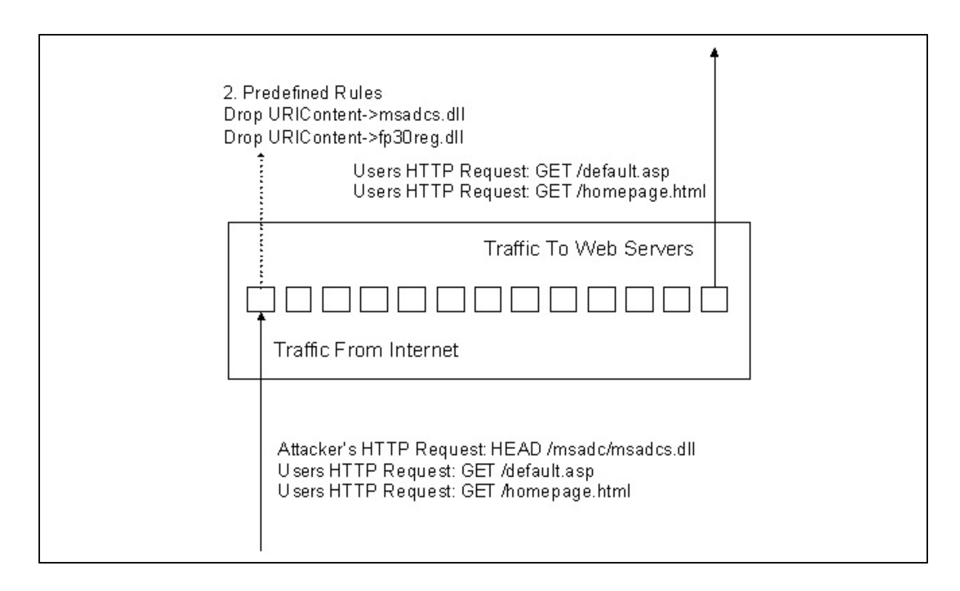


FIG 3-16: Ejemplos de IPS layer 7 switch.

Como los otros sistemas de prevención de intrusos, es capaz de filtrar el tráfico, buscar patrones en la red y controlar las conexiones de entrada y salida a los servidores. A diferencia de los sistemas basados únicamente en hardware, son capaces de controlar eficientemente el ancho de banda utilizado en cada uno de los servidores y variarlo en función de las necesidades de cada momento.

3. **Application firewall/IDS**: Este grupo de IPS son programas autónomos que corren en cada uno de los servidores a proteger. De esta forma, se encargan de proteger únicamente un ordenador o servicio concreto, y no una red o conjunto de ordenadores.

La sobrecarga (*overhead*) de proceso que producen en el ordenador, queda compensada por su gran capacidad de configuración, lo que permite que se centre únicamente en las partes o servicios que nos interese proteger (por ejemplo WWW) en lugar de controlar todo el sistema (ver figura 3-17).

El sistema IPS crea unos perfiles (*profile*) para cada uno de los servicios a monitorizar de forma que podemos ajustar de manera muy específica los parámetros de control minimizando la sobrecarga.

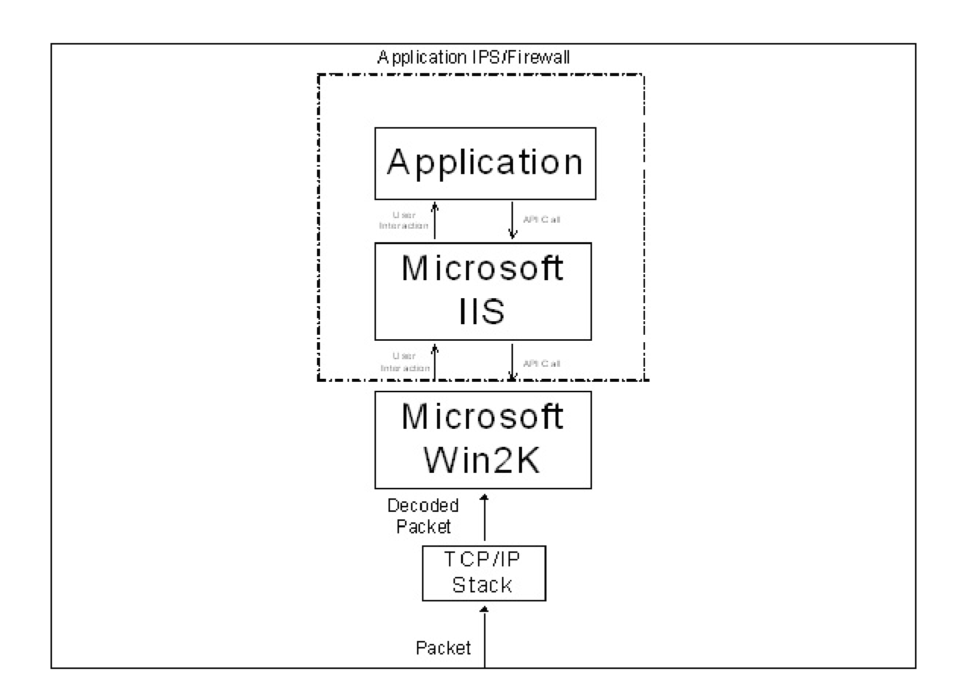


FIG 3-17: Ejemplos de IPS application firewall/IDS.

La aplicación IPS se coloca entre el gestor de comunicaciones del sistema operativo y la aplicación, monitorizando todo el tráfico y efectuando todas las acciones que considere necesarias (*forward, drop, log, unlog, rewrite*) según la configuración creada (perfil).

Hay que tener en cuenta que debemos ir sincronizando el perfil con los distintos cambios o actualizaciones de la aplicación que se vayan produciendo, ya que el IPS puede detectarlo como un intruso y cesar el servicio.

4. **Hybrid switches**: Estos sistemas de prevención de intrusos se basan en combinar una parte de software y otra de hardware.

   La parte hardware es la encargada de filtrar el tráfico en tiempo real (debido a su mayor velocidad de proceso), limitar el numero de peticiones a los servidores y regular los anchos de banda de entrada y salida adecuándolos a las necesidades y demandas en cada momento.

   La parte software se instala en cada uno de los servidores a monitorizar de forma que se crea un perfil específico para cada una de los servicios. Este componente se comunica con la parte hardware para conseguir un comportamiento mas adecuado a las necesidades reales. Por otro lado, también realiza parte del filtrado para descargar el hardware y eliminar los cuellos de botella.

5. **Deceptive applications**: Este tipo de aplicaciones empezaron a utilizarse en el 98 y se basa en una aproximación diferente y más proactiva frente a la detección de intrusos. Su aplicación se divide en tres fases (ver figura 3-18):

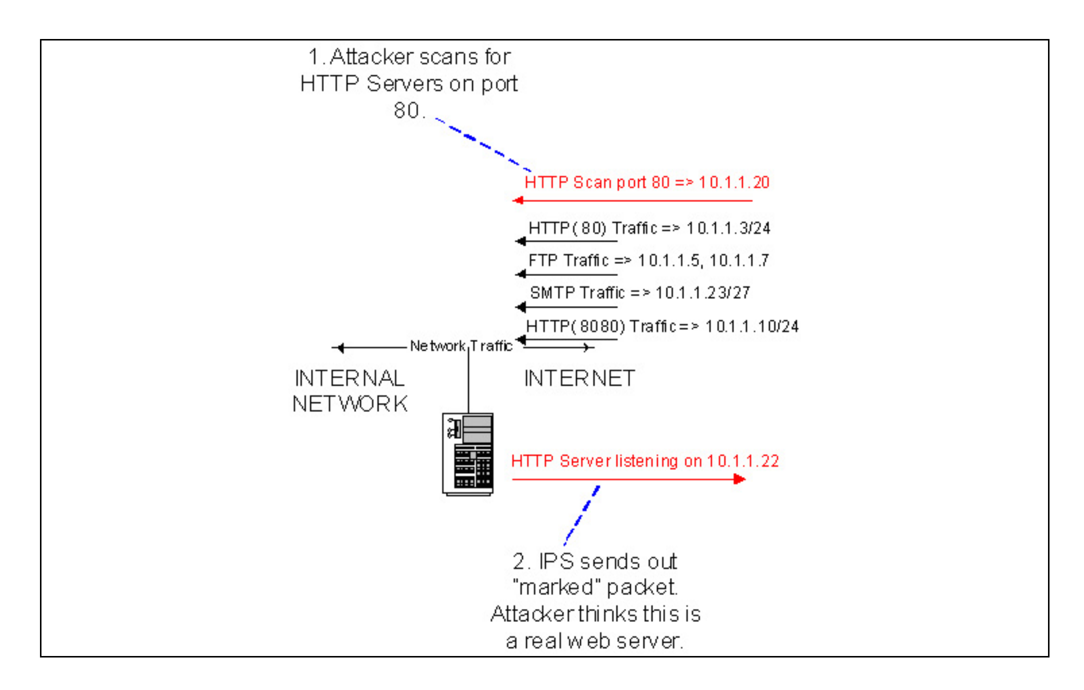


FIG 3-18: Deceptive applications.

En una primera fase se analiza todo el tráfico de la red con el objetivo de crear un modelo que represente todo el tráfico "normal" que circula por la red.

Después (opcionalmente, aunque se recomienda) se retoca el modelo manualmente por parte del administrador del sistema para adecuarlo a la realidad de la red (*profiling*). Finalmente el sistema asignará unos pesos a cada uno de los distintos eventos que ha observado en el análisis.

En una segunda fase, el sistema monitoriza el tráfico y las peticiones que circulan por la red. Cuando observa peticiones a servicios que no existen o que no están disponibles, reacciona enviando como respuesta un paquete "marcado" simulando la existencia del servicio y anotando los parámetros de origen de la petición.

La tercera fase se produce cuando el atacante utiliza los datos obtenidos en incursiones anteriores a la red (fase 2) para lanzar un ataque sobre servidores o servicios. En este momento el sistema reconoce al atacante y bloquea su acceso a la red.

3.10 Limitaciones de los IDS

Después de analizar los distintos tipos, arquitecturas y características de los sistemas de detección de intrusos, comentaremos brevemente que aspectos dejan más abiertos o no resueltos totalmente en la actualidad.

Los sistemas NIDS se basan en la observación del tráfico que circula por la red, esta única fuente de información le confiere una serie de limitaciones:

- Si alguien crea una puerta trasera ("*back-door*") o crea una red paralela en la intranet, probablemente los IDS no lo detectarán puesto que están pensados y configurados para el rango IP señalado.

Si alguien se apropia de una cuenta y se autentica con el *login* y *password* correcto, tampoco.

- Si los sensores que alimentan al IDS o el propio IDS no funciona (la máquina, el sistema operativo o el programa no se encuentra operativo) no observaremos nada.

  Es importante realizar un seguimiento periódico de la actividad de estos programas, ya que el hecho de tenerlos instalados o hacer un simple *ping* a la máquina para ver si está operativa no es una política de seguridad aceptable.

- Muchos IDS simplemente soportan protocolos muy extendidos (usualmente basados en IP como FTP, HTTP...) pero no protocolos de usos minoritarios comos el IPX/SPX o SNA. Existen muchos "*exploits*" disponibles para estos protocolos que pasarán inadvertidos en nuestra red. Si utilizamos Novell, NetBIOS, SNA... debemos tener en cuenta este punto.

- Los IDS tienen un máximo de capacidad de proceso, por lo que en momentos de mucho tráfico suelen descartar los paquetes que no pueden procesar.

  Raramente un IDS avisa de este punto, con lo que es conveniente calibrar periódicamente si nuestro IDS es capaz de soportar realmente toda la carga de la red (accesos a discos, swap, filtros muy complicados...). No hay que olvidar que usualmente un IDS no es más que un programa que funciona en un ordenador.

Por otro lado, un IDS basado únicamente en la búsqueda de firmas (al igual que muchos antivirus) presenta dos grandes problemas:

1. El número de firmas va aumentando cada año (actualmente existen más de 20.000 para los antivirus) y tan sólo refleja los ataques que han sido ya identificados y analizados (UC Davis nos da una muestra de firmas para IDS en [WWW67]).

En los informes de ataques recibidos, siempre salen reflejados los mismos ataques que en el resto de Internet. Esto es debido a que los nuevos ataques aún no han sido detectados y documentados, con lo que no existe una firma a la que asociar el ataque recibido, lo que puede crear una falsa sensación de seguridad al ver que detectamos siempre los mismos ataques.

2. En el caso de una red con cientos de ordenadores, todo y detectar ataques ya conocidos con nuestro IDS ¿podemos estar seguros que todos nuestros ordenadores son invulnerables a este ataque?.

   Por otro lado el responsable de los ataques muy conocidos que detecta nuestro IDS puede ser ignorado como un aprendiz de hacker (*script-kiddie*). ¿Quien nos garantiza que no irá "in crescendo" probando otros ataques no documentados que no detectemos?.

El uso de otras técnicas de seguridad que complementen a los sistemas de detección de intrusos (como los Honeypots y Honeynets) es vital para conseguir un sistema de seguridad fiable y eficaz.

3.11 El ataque Mitnick

En este último apartado del capítulo realizaremos un breve pero completo análisis a un ataque considerado como el umbral mínimo de detección por un sistema IDS actual para considerarse una herramienta útil y fiable [Nor99].

Kevin Mitnick es el hacker mas famoso de todos los tiempos [WWW103][WWW104], fue capturado por Tsutomu Shimomura y entregado al FBI. Fue condenado a 68 meses de prisión de los que sólo cumplió 60. Está en libertad desde el 21 de enero de 2000.

El ataque de Kevin Mitnik se basa en el conocimiento profundo de los protocolos de Internet (TCP, UDP, OC, ICMP...) y la combinación de dos técnicas de hacking [Nor99]:

- **SYN Flooding**: Esta técnica consiste en realizar miles de peticiones de conexión a un ordenador sin finalizar completamente ninguna de ellas. De esta forma, se intenta bloquear a la máquina que recibe las peticiones consumiendo todos sus recursos de red (para mas información ver el capítulo de ataques DDOS).

- **TCP Hijacking**: Esta técnica consiste en conseguir apropiarse de una conexión ya iniciada, desviándola hacia el ordenador del atacante (para mas información ver el capítulo de Redes IP).

El objetivo de nuestro ataque es el de conseguir acceso a una máquina a la cual legalmente no podemos entrar. La manera de conseguirlo consiste primero en detectar las conexiones existentes en la máquina que queremos atacar. Posteriormente seleccionamos a una víctima y le robamos la conexión que tenía establecida. De esta forma, conseguimos un acceso a la máquina sin necesidad de validarnos en el sistema (login).

Kevin Mitnick utilizó las relaciones de confianza (*trust relationships*) entre el cliente autorizado y la máquina a atacar para robar un acceso una vez que este se hubiera autenticado en el sistema.

Para ello, primero se instaló un programa de tipo "*sniffer*" (**tcpdump** por ejemplo) para examinar el tráfico de la red dónde está situado el ordenador atacado. Se supone que también se usaron otras técnicas standard para la obtención de información del ordenador a atacar (consultas a los *daemons* de servicios de finger, NFS, RPCinfo...)

Observando el tráfico (ver figura 3-19) podemos deducir (predecir mas concretamente) que cada vez que se inicia una petición de conexión a "*x-terminal*" (servicio de *shell* remoto bajo X-Windows), la respuesta del servidor contiene un número de secuencia 128.000 unidades mayor que el enviado en la petición.

IMPORTANTE: este número NO es siempre el mismo y depende del programador del software de red de cada sistema.

1. Primer paso del protocolo de tres pasos para establecimiento de conexión TCP (three-way handshake[ℜ])

14:18:25.906002 apollo.it.luc.edu.1000 > x-terminal.shell: S **1382726990**:1382726990 (0) win 4096

2. Segundo paso. El servidor nos da su número de secuencia y el ACK del nuestro + 1.

14:18:26:094731 x-erminal.shell > apollo.it.luc.edu.1000: S **2021824000**:2021824000 (0) ACK **1382726991** win 4096

3. Tercer paso. El cliente envía su número de secuencia y el ACK del número del servidor + 1

14:18:26.172394 apollo.it.luc.edu > x-terminal.shell: S 1382726991: 2021824001 (0) win 0 ACK **2021824001**

En sucesiones peticiones de x-term observamos (extracto del tráfico):

14:31:25.906002 apollo.it.luc.edu.1000 > x-terminal.shell: S **1382726992**:1382726992 (0) win 4096
14:31:26:094731 x-erminal.shell > apollo.it.luc.edu.998: S **2021952000**:2021952000 (0) ACK **1382726993** win 4096
14:31:26.172394 apollo.it.luc.edu > x-terminal.shell: S 1382726993:2021952001 (0) win 0 ACK **2021952001**
....

De lo que podemos deducir (predecir) que el número de secuencia enviado por el servidor es siempre 128.000 unidades superior al anterior.

2021952000 - 2021824000 = **128.000**

FIG 3-19: Establecimientos de conexión observados por Mitnick.

La obtención de los números de secuencia es vital, ya que en el caso de enviar un paquete con un número de secuencia erróneo, automáticamente se genera un RESET y el servidor cierra la conexión.

Una vez deducido el número de secuencia que obtendremos, debemos proceder a "secuestrar" la conexión del usuario correctamente autenticado. Esto fue posible debido a que una vez autenticada la comunicación (dirección IP de origen + puerto) <=> (dirección IP de destino + puerto) esta no vuelve a ser verificada por el servidor.

La dirección IP se encuentra en la cabecera del paquete IP mientras que los números de secuencia se encuentran en la cabecera TCP. Las aplicaciones simplemente mantienen el número de secuencia para asegurarse que el datagrama recibido pertenece a la

[ℜ] En el primer capítulo está ampliamente explicado el protocolo de tres pasos para el establecimiento de conexiones TCP.

secuencia correcta (de aquí la importancia de poder predecir el número de secuencia enviado por el servidor).

De esta forma, una vez iniciada la conexión desde un cliente autorizado (paso 1 de la figura 3-18), Mitnick "silencia" al cliente mediante un ataque SYN FLOOD [Tan03].

Simultáneamente el servidor completa el segundo paso de la conexión (paso 2 de la figura 3-18) que nunca llegará a su destino, puesto que el cliente ya ha sido silenciado.

A continuación, Mitnick "crea" manualmente un paquete IP que contiene el número de secuencia del servidor + 1 (ACK predecible) falseando la dirección de origen por la del cliente autorizado.

En este punto la respuesta del servidor (paso 3 de la figura 3-18) **NO** es vista por Mitnick, ya que al falsear la dirección de origen nunca le llegará la respuesta. Se "confía" en que este paquete del servidor se ha enviado y que el ataque de SYN FLOOD sea exitoso y silencie al cliente autorizado, ya que si no respondería a esta petición con un RESET.

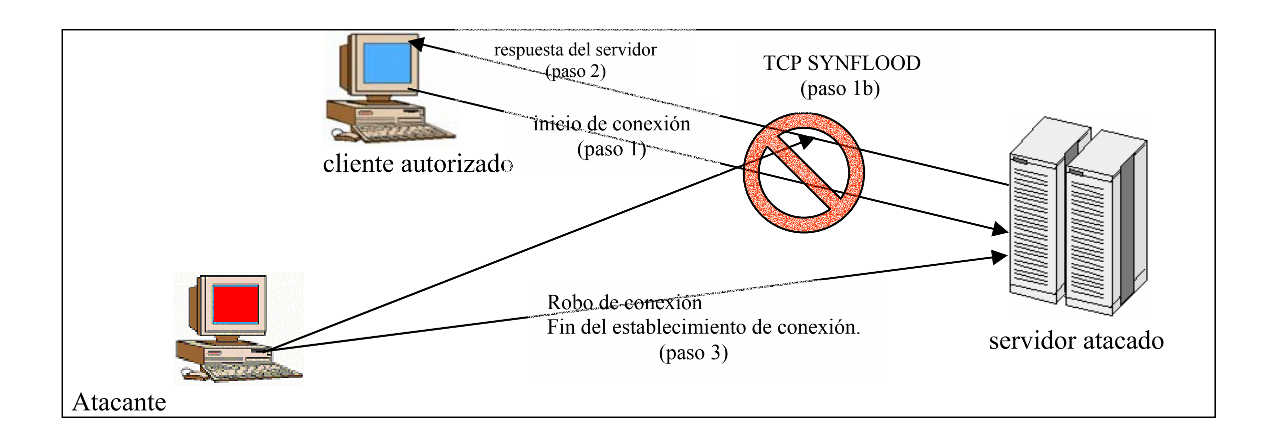


FIG 3-18: Ataque de Kevin Mitnick.

Una vez conocidos y validados los números de secuencia de la conexión TCP, el atacante ejecuta el siguiente comando [WWW106]:

rsh x-terminal "echo + + >>/.rhosts

Que permite a cualquier usuario iniciar una conexión X-terminal como root desde cualquier máquina. Una vez llegado a este extremo, el atacante tiene acceso total a la máquina.

3.12 RESUMEN

En este capítulo dedicado a los sistemas de detección de intrusos o IDS, hemos definido los conceptos de IDS, NIDS y IPS. Se han definido las distintas arquitecturas o configuraciones que forman cada uno de estos sistemas.

Se ha comentado la importancia de introducir filtros y firmas que nos permitan evitar el imposible trabajo de filtrar todo el tráfico de la red en tiempo real reduciéndolo al análisis de los eventos de interés. También se han comentado los distintos standares propuestos para la comunicación de los distintos sistemas NIDS (CIDF, CIDF, OPSEC...).

Además se han descrito las problemáticas asociadas al análisis de los datos obtenidos por los sistemas de detección de intrusos y las consecuencias asociadas (falsos positivos y falsos negativos). También se han comentado los sistemas de prevención de intrusos (IPS), que se vislumbran como una de las posibles líneas de evolución de los sistemas NIDS.

Finalmente se realiza una explicación del famoso ataque de Kevin Mitnick, considerado el umbral mínimo de detección para un sistema IDS.

CAPITULO 4

Honeypots y Honeynets

En los capítulos anteriores hemos examinado las distintas técnicas de ataques de denegación de servicio DOS/DDOS así como los sistemas de detección de intrusos (IDS). En este capítulo plantearemos el cambio de escenario que están sufriendo las comunicaciones por Internet debido al rápido avance tecnológico y cómo estas han ido modificando los ataques a las redes de ordenadores.

Nuevos tipos de ataques requieren nuevos modelos que permitan ampliar el espectro de seguridad cubierto anteriormente. Describiremos en profundidad las características, tipos y ubicaciones de los Honeypots. Estos son un intento de conocer al enemigo "desde dentro", conocer sus técnicas y motivaciones proporcionándole un entorno controlado pero interactivo en el que pueda "jugar" mientras es espiado.

Comentaremos los distintos tipos, sus arquitecturas así como las posibles ubicaciones que permiten y las implicaciones legales que entraña su uso.

Posteriormente nos centraremos en las Honeynets, que aparecen como desarrollo del concepto de Honeypot. Se explicarán sus distintas características y configuraciones (generaciones) y veremos como permiten la implementación de completos sistemas para el estudio de atacantes.

Finalmente se describirán las Honeynets virtuales que permiten mediante un uso mínimo de recursos una implementación total de una o varias Honeynets.

4.1 Nuevos escenarios de ataques

En los capítulos anteriores hemos analizado los escenarios típicos generadores de amenazas para cualquier sistema conectado a Internet. Con el aumento de los anchos de banda disponibles y el abaratamiento de los accesos a la red hemos podido observar una evolución de las técnicas de ataques existentes en la actualidad.

Anteriormente los ataques se basaban en razonamientos simples, dada una máquina/dominio/servicio conocido (por ejemplo el de correo personal gratuito de Hotmail → www.hotmail.com), bastaba con obtener su dirección IP (vía consulta DNS por ejemplo) y proceder con cualquiera de los ataques anteriormente descritos.

Este procedimiento que consideraremos "standard" circunscribía los destinatarios de ataques informáticos a grandes empresas, organismos oficiales y universidades. Sin embargo, la mejora de la conectividad nos permite ahora realizar combinaciones de ataques que hasta hace unos pocos años eran imposibles.

La realización de un análisis completo (*scan o probe*) de toda una clase A, B o C (ver figura 4-1) deja de ser una utopía para convertirse en una simple cuestión de días o incluso horas.

CLASE	RANGO		ORDENADORES
A	0.0.0.0	127.255.255.255	128 redes de 2^24 direcciones
B	128.0.0.0	191.255.255.255	2^14 redes de 2^16 direcciones
C	192.0.0.0	223.255.255.255	2^21 redes de 2^8 direcciones

FIG. 4-1: Redes y direcciones IP en Internet.

La proliferación y difusión de herramientas específicas para este tipo de comportamientos (podemos encontrar cientos de servidores WWW con unas simple búsqueda en Google), ha ido pareja al aumento de jóvenes con ganas de emular las películas de piratas informáticos.

Precisamente esta gran cantidad de público ha llevado a la creación de herramientas con interfaces (*front-ends*) muy sencillos y amigables que permiten prácticamente a cualquier persona sin muchos conocimientos (*script-kiddies*) rellenar un par de campos de parámetros (por ejemplo la dirección IP de inicio y final) y lanzar ataques indiscriminados.

Muchas herramientas simplemente comprueban si el ordenador atacado presenta una vulnerabilidad o versión antigua de software, cosa que antes o después les llevará a conseguir acceso a algún sistema conectado a Internet.

La proliferación de este tipo de herramientas junto a pensamientos erróneos del tipo "nadie me conoce" o "quien va a querer atacarme, no merece la pena" lleva a que el número de ordenadores comprometidos sea muy difícil de aproximar, y únicamente cuando es demasiado tarde (el ordenador ya ha sido utilizado para un ataque o han borrado información) las prisas y necesidades de seguridad en la empresas pasan a tomarse en serio.

Podemos observar como en el último trimestre de 2002 [Gre03] mas del 73% de los ataques observados en Internet pertenecen a la categoría de "**actividad sospechosa en la red**", indicando una bajada importante de los ataques directos y un gran aumento de comportamientos poco claros o sospechosos en la red.

Estos barridos "**ciegos**" por Internet tienen por objetivo la creación de listas de ordenadores conectados a la red (*shopping lists*). Posteriormente, estas listas que contienen las direcciones IP son utilizas por programas mas sofisticados que comprueban los servicios existentes en las máquinas buscando alguna vulnerabilidad que pueda ser aprovechada para obtener el control del ordenador.

Una vez conseguido el acceso se suelen instalar programas de puerta trasera (*back door*) para poder acceder de forma invisible a la máquina. Esta queda en estado de "espera" por tiempo indefinido hasta que el atacante (*hacker*) desee hacer uso de ella.

Cabe señalar también que muchas veces el acceso a un ordenador es utilizado como trampolín para efectuar nuevos barridos de forma que el verdadero origen del barrido no queda comprometido. Además, al igual que en los ataques DDOS, mas ordenadores comprometidos conectados a Internet significa mas potencia de búsqueda de posibles nuevas víctimas.

4.2 Historia de los Honeypots

Una vez definido el nuevo escenario hacia el que Internet se encamina, podemos observar como muchas de las premisas clásicas de seguridad ("si nadie conoce mi red nadie puede encontrarla" o "cuando menos documentada esté mi estructura más difícil será para un atacante el poder entrar") dejan de tener sentido.

Así mismo, las herramientas típicas de seguridad por excelencia (firewalls, IDS) dejan un hueco cada vez más importante sin cubrir.

Históricamente las primeras referencias a un sistema de monitorización de intrusos aparecen ya en la bibliografía sobre los años 90 de la mano de Clifford Stoll [Sto90], sin embargo los líderes en la investigación y desarrollo del concepto de Honeypot se agrupan en el **HoneyNet Project** [WWW119][WWW149].

Este proyecto se inició informalmente en la lista de correo "*Wargames*" en abril de 1999 [Kue01] gracias a los correos cruzados entre varios expertos en seguridad de redes que culminaron con el desarrollo formal del proyecto antes de finalizar el año.

En junio de 2000 y por espacio de tres semanas, el Honeypot del proyecto fue atacado y comprometido por un famoso grupo de hackers, lo que permitió el estudio del comportamiento de este grupo en "real" así como demostrar la viabilidad y utilidad de esta nueva herramienta de seguridad.

Este conocido incidente catapultó mediáticamente el concepto de Honeypot como la última tendencia en seguridad de redes convirtiendo su libro en un best-seller de lectura obligatoria para todos los profesionales de la seguridad [Hon01].

A inicios de 2001 se convirtió en una organización si ánimo de lucro [WWW123] dedicada al estudio de los hackers (*blackhats*) que actualmente está compuesta por más de 30 miembros permanentes.

4.3 Honeypots

El concepto de *Honeypot* no fue extraído o inventado de la nada, sino que es fruto de la realización de varios estudios en el campo de la seguridad de redes de ordenadores [Sch99][Hon01][Mcm01][Kue02][Ran02][VP02][Lev03].

Definiremos **Honeypot** (tarro de miel textualmente) como "*un recurso de red destinado a ser atacado o comprometido. De esta forma, un Honeypot será examinado, atacado y probablemente comprometido por cualquier atacante. Los Honeypot no tienen en ningún caso la finalidad de resolver o arreglar fallos de seguridad en nuestra red. Son los encargados de proporcionarnos información valiosa sobre los posibles atacantes en potencia a nuestra red antes de que comprometan sistemas reales.*" [Spit02].

Esta nueva aproximación a la seguridad de redes de ordenadores rompe muchos tabúes clásicos que se daban como axiomas en la seguridad informática "clásica":

- Por un lado, este nuevo elemento no sirve para eliminar o corregir fallos de seguridad existentes en nuestra red. Si nuestra red es vulnerable, añadir un Honeypot no solventará este fallo.

- Por otro lado, en lugar de evitar a cualquier precio que un atacante fije su interés en nuestra red, le invitamos o incitamos (para ser exactos deberíamos decir le permitimos) a que entre y ataque nuestra red.

Este interés en dejar una puerta abierta hacia Internet puede considerarse temeraria o incluso suicida. Si ya existen cientos de miles de ataques a sistemas "seguros" ¿porqué dejar un pote de miel en medio del camino del "oso"? ¿Por qué incitar a que ataquen nuestro sistema cuando nuestro objetivo es conseguir exactamente lo contrario?

Por muy eficientes que seamos en nuestro trabajo como administradores de red, es imposible mantener todos nuestros sistemas al día (*up to date*). Cada día se descubren al menos una docena de nuevos fallos (*bugs*) en el software existente [WWW21] (sistemas operativos, servidores de aplicaciones, servidores WWW...), consecuentemente de una forma regular salen nuevos parches y actualizaciones para todo tipo de software (ver figura 4-2).

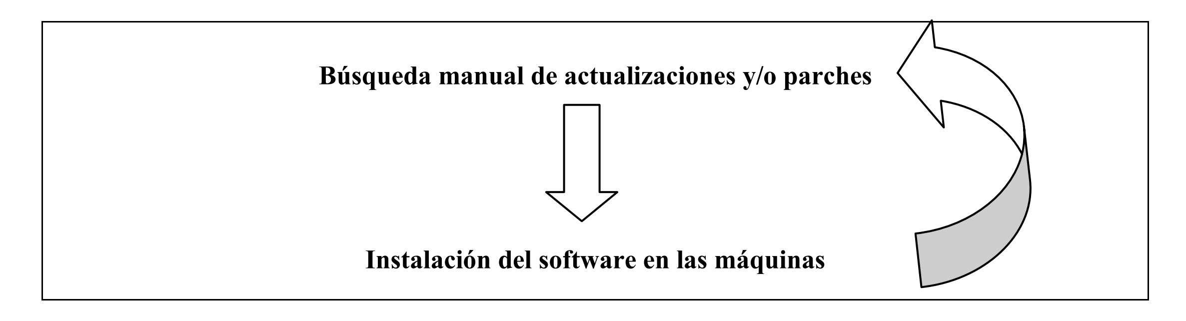


FIG. 4-2: Ciclo de actualización del software.

De esta forma, podemos observar como en la realidad es totalmente imposible mantener "controlada" una red con cientos de ordenadores y usuarios.

Si nosotros mismos examinamos periódicamente nuestra red como si fuéramos un intruso, es decir, examinando todos los ordenadores de la red en búsqueda de los servicios disponibles, podemos modificar el ciclo de actualizaciones de software (ver figura 4-3).

Así mismo, podemos por un lado corregir los problemas de seguridad en nuestras maquinas de producción (aplicación de parches de seguridad y actualizaciones de los productos comerciales) y por otro mantener "**aparentemente**" las vulnerabilidades corregidas de forma que nos permitan descubrir a potenciales atacantes.

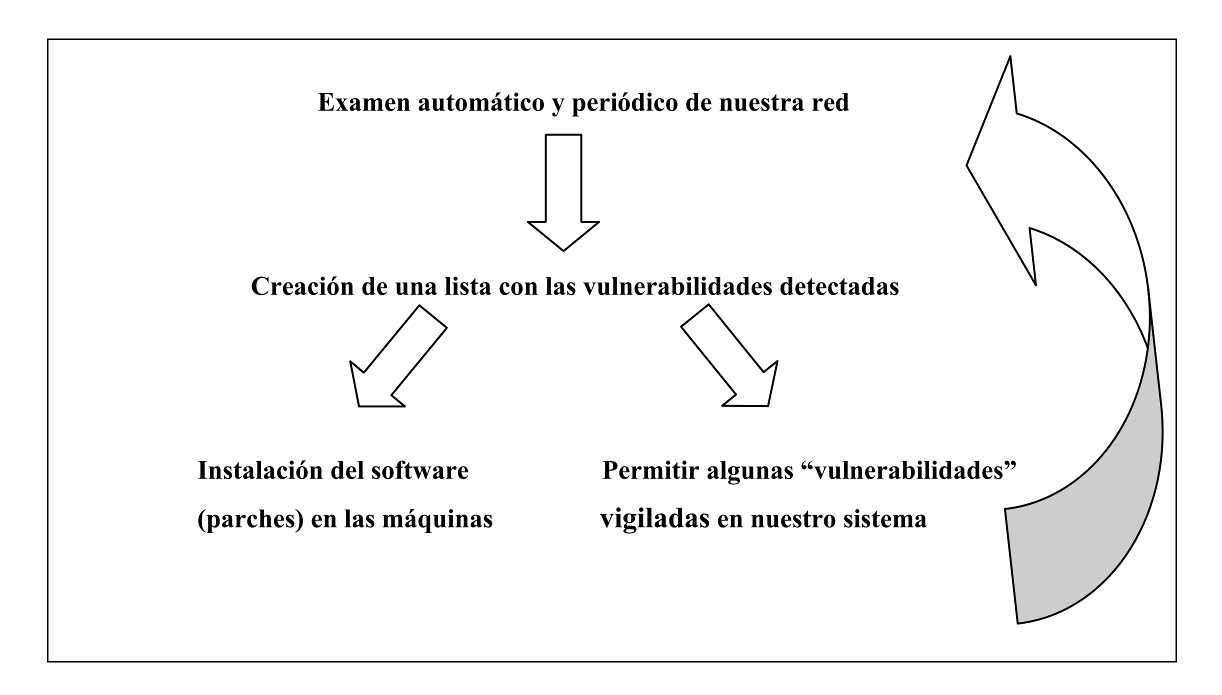


FIG. 4-3: Ciclo de actualización del software "modificado".

Estas vulnerabilidades son "mantenidas" expresamente en máquinas controladas (*honeypots*) por nuestros sistemas de seguridad (firewalls, IDS…).

De esta forma, desde el punto de vista de un atacante tiene ante sí un sistema candidato a ser explotado, por lo que invertirá más tiempo y herramientas menos sofisticadas (ya que el sistema es vulnerable por construcción) [Spi02][Spi03-2][Hon01][WWW122] que facilitarán su detección por nuestros sistemas.

Por otro lado, la inclusión de técnicas de engaño [Sch99] como el famoso "*Deception Toolkit*" [WWW134] evita que señalemos a los atacantes como y de que forma exacta pueden hacernos daño.

Si nuestra política es únicamente la de mantener nuestro sistema parcheado y al día (figura 4-2), cuando un atacante detecte una vulnerabilidad en nuestro sistema estamos perdidos, puesto que la desconocemos y esta existe realmente!.

Si mantenemos distintas vulnerabilidades de forma "controlada", estamos ofreciendo información confusa/engañosa al atacante, cosa que dificultará su trabajo. Así mismo, la detección de una vulnerabilidad en nuestro sistema no tiene porque implicar una caída del mismo, al contrario, nos permitirá descubrir a un atacante en potencia y tomar las medidas oportunas antes de que entre realmente en nuestro sistema.

Este enfoque nos permitirá obtener más información sobre el atacante que podremos utilizar en su contra. Además, este tipo de paradigma es válido tanto para atacantes internos como externos, que al intentar extender sus ataques a más ordenadores de la red, comprobarán que:

a) El resto de los ordenadores de nuestra red NO son vulnerables a los métodos que utiliza, obligándole a un cambio de estrategia que suele ser un abandono en búsqueda de otro sistema más sencillo de atacar.

b) El sistema informático de nuestra red es proactivo y reacciona a los accesos del atacante limitando su alcance.

4.3.1 Valor añadido

Una vez expuesto el nuevo ámbito de ataques en Internet y como este nos conducía hacia el concepto de Honeypot, pasamos a enumerar las principales características y ventajas que nos ofrecen los sistemas basados en Honeypots [SR01]:

- **Genera un volumen pequeño de datos:** Al contrario que los sistemas clásicos de seguridad (Firewalls, IDS…) que generan cientos de megas de ficheros de logs con todo tipo de información "poco útil", los Honeypots generan muy pocos datos y de altísimo valor.

  Los Honeypots son ordenadores "expresos" por lo que ningún usuario/sistema "normal" debe acceder a ellos. De esta forma, cualquier acceso nos revelará un atacante o una configuración errónea de un sistema. **No existen los falsos positivos.**

- **Necesita unos recursos mínimos:** A diferencia de otros sistemas de seguridad, las necesidades de un Honeypot son mínimas. No consume ni ancho de banda ni memoria o CPU extra. No necesita complejas arquitecturas o varios ordenadores centralizados, cualquier ordenador conectado a la red puede realizar este trabajo.

- **Universalidad:** Este tipo de sistemas sirven tanto para posibles atacantes internos como externos. De esta forma, obviamente, se ha de evitar poner a las máquinas nombres como "honeypot" o "attack-me" (muchas veces ni tan siquiera están dadas de altas en los DNS). Su objetivo es pasar desapercibidas en una red como una máquina más.

Por otro lado, y como siempre suele ocurrir, todo sistema tiene también unas contrapartidas o desventajas asociadas. En el caso de los Honeypots los principales inconvenientes son:

- Son elementos totalmente pasivos. De esta forma, si no reciben ningún ataque no sirven de nada.

- Son fuentes potenciales de riesgo para nuestra red. Debido a la atracción que ejercen sobre posibles atacantes, si no calibramos perfectamente el alcance del Honeypot y lo convertimos en un entorno controlado y cerrado (*jailed environment*) , puede ser utilizado como fuente de ataques a otras redes o incluso a la nuestra propia

- Consumen una dirección IP como mínimo. De todas formas, este inconveniente es mínimo, ya que lo ideal es asignar direcciones IP del rango de direcciones libres.

Del análisis de todas las características principales de los Honeypots y aplicando el desglose del concepto de seguridad [Sch00] en prevención, detección y reacción (ver figura 4-4) obtenemos el siguiente análisis:

1. Los Honeypots tienen un limitado carácter **preventivo**. No evitarán o disuadirán a ningún posible atacante.

2. Tienen un alto grado de **detección**. Si bien son elementos pasivos, los atacantes rara vez se centran en una simple máquina, sino que buscan por toda la red posibles víctimas, lo que hace que antes o después se encuentre con el Honeypot.

3. La **reacción** es otro de los valores que añade el uso de Honeypots. En los de Honeypots de producción se puede de forma automática generar los comandos necesarios para evitar el acceso del atacante al resto del sistema. En los de investigación, además nos permiten a posteriori la ejecución de técnicas forenses (*computer forense*) para examinar el comportamiento del atacante y descubrir sus comportamientos.

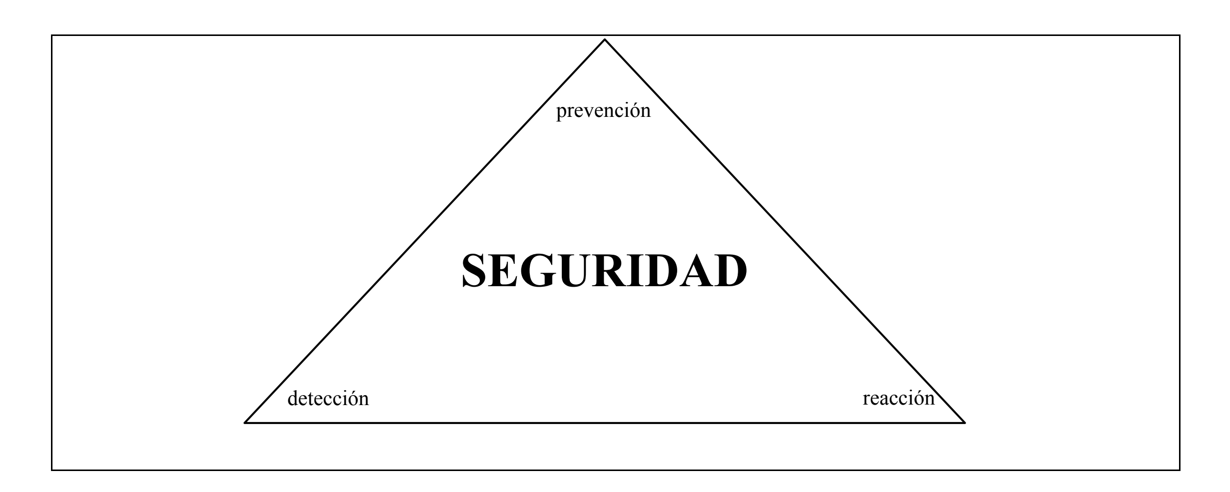


FIG. 4-4: Concepto de seguridad: prevención, detección y reacción.

4.3.2 Clasificación

La taxonomía de los diferentes tipos de Honeypots depende de la bibliografía consultada puesto que como todo nuevo concepto, su estandarización es compleja y aún no ha sido universalmente aceptada.

En este capítulo citaremos las dos clasificaciones mas aceptadas por la comunidad, ya que en el momento de la realización de este trabajo no se ha observado una imposición de una sobre la otra.

La primera clasificación presentada se basa en la funcionalidad que se desea asignar al Honeypot [Cho01][SR01][Ran02][Spi03] y [WWW130]:

- **Honeypot de producción** (*Production Honeypot System*): Su principal objetivo es el de mitigar el riesgo de un ataque informático a la red de una institución o empresa. De esta forma, un Honeypot de producción simula diferentes servicios con el único objetivo de ser atacado.

  Una vez descubierto el atacante/s, se "avisa" al resto del sistema para que tome las medidas oportunas (denegar cualquier acceso con un origen determinado, limitar las capacidades de un servicio, paralizar varios servicios momentáneamente…).

- **Honeypot de investigación** (*Research Honeypot System*): Su principal objetivo es el de recoger información sobre los distintos atacantes así como de sus comportamientos y técnicas asociados. También han sido diseñadas para ser comprometidas al igual que los de producción, sin embargo no añaden ninguna capacidad extra de seguridad o mitigación de los ataques.

  Suelen ofrecer servicios reales (no los simula) e incluso pueden llegar a permitir que el atacante tome el control total de la máquina (acceso *root*).

Otra posible clasificación de los Honeypots hace referencia al grado de compromiso o riesgo que esta introduzca en nuestra red [BP02][Spi02]:

- **Compromiso bajo** (*Low involvement*): El sistema Honeypot simplemente simula la existencia de que existe algún servicio común (WWW, FTP, TELNET...) escucha y almacena todas las peticiones recibidas en ficheros de logs. De esta forma, tenemos un sistema totalmente pasivo que simplemente registra peticiones de acceso ya que NO respondemos a ninguna de ellas o interaccionamos con el atacante.

  En el caso de un servidor TELNET, podríamos contestar a las peticiones TCP del puerto 23 permitiendo un establecimiento de conexión, pero cerrándola inmediatamente después del tercer paso (ver capítulo 1) evitando cualquier posibilidad de que el usuario pueda conectarse al sistema (envío y recepción de login y password).

  El riesgo que introduce esta variante es mínimo puesto que el atacante nunca podrá acceder a la máquina, lo que nos hace perder la posibilidad de investigar y analizar sus técnicas (ver figura 4-5).

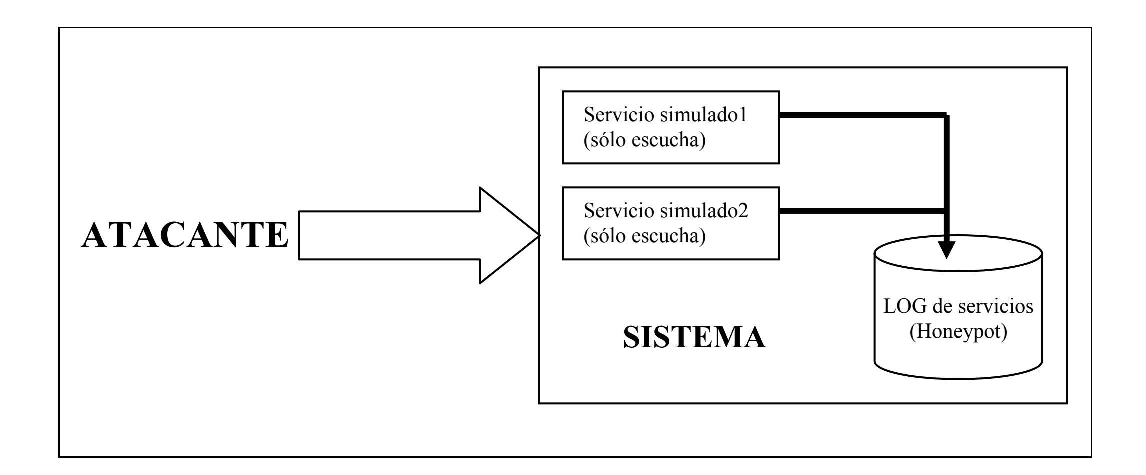


FIG. 4-5: Honeypots de compromiso bajo.

- **Compromiso medio** (*Medium involvement*): En este grupo los sistemas simulan la existencia de uno o varios servicios de forma más sofisticada. Con este tipo de Honeypots se pretende captar la atención del atacante y permitir un grado mayor de interacción que nos permitirá analizar minimamente el comportamiento del atacante.

  El grado de riesgo aumenta moderadamente, ya que por un lado el servicio sigue siendo una simulación, lo que permite tener acotado/enjaulado la interacción entre el atacante y el servicio. Por otro lado, si existe un fallo en la implementación del servicio simulado, el atacante puede aprovecharlo para atacar el sistema real[ℜ] (ver figura 4-6).

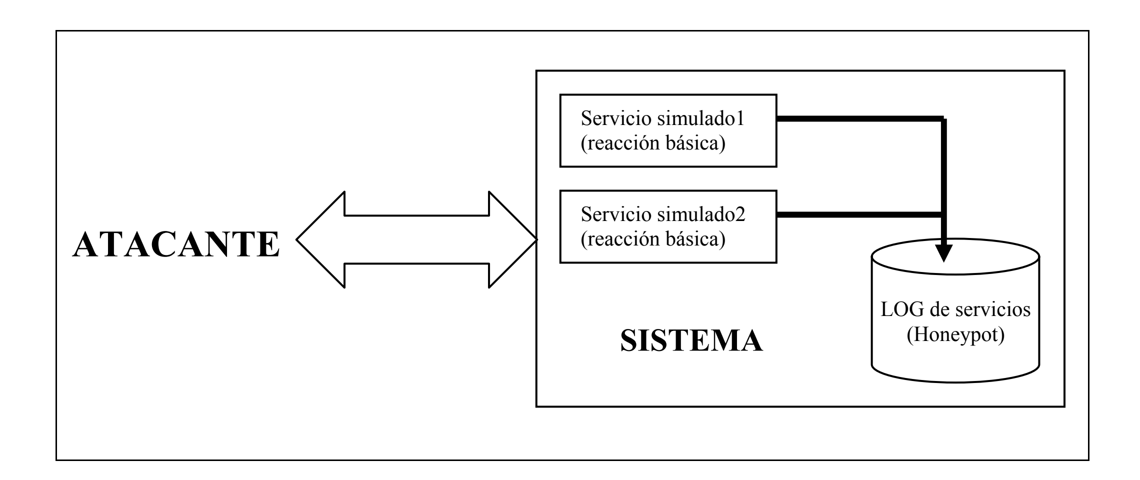


FIG. 4-6: Honeypots de compromiso medio.

- **Compromiso alto** (*High involvement*): En este grupo se encuentran aquellos sistemas que no simulan diferentes servicios, sino que utilizan un entorno real con servicios de verdad (*in the wild*).

  Este tipo de Honeypots son muy atractivas para los atacantes y permiten un estudio completo de su comportamiento. Deben estar constantemente monitorizadas ya que su peligro consiste en que si un atacante logra acceso a ella puede disponer de todo el sistema como le plazca.

[ℜ] Generalmente, estos programas son más seguros que lo servicios que simulan.

Esto significa que ya no podemos considerarlo como un lugar con logs "fiables" y puede ser utilizado para atacar otros sistemas de nuestra red o incluso de otras conectadas a Internet (ver figura 4-7).

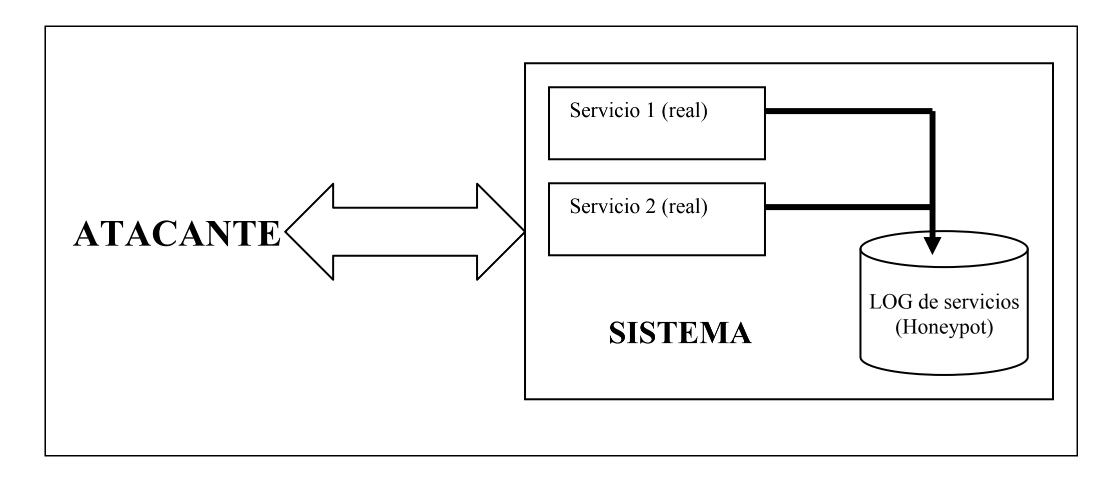


FIG. 4-7: Honeypots de compromiso alto.

En Internet podemos encontrar varias herramientas que implementan los distintos tipos de Honeypots comentados antes [WWW122][WWW123][WWW124][WWW134] y [Mar01][BP02]. Estas herramientas son muy flexibles y permiten la total configuración de servicios e incluso las vulnerabilidades a simular. Además, algunos incluso permiten escoger el tipo de sistema operativo y versión que se desea simular (Solaris, Linux, Windows…).

4.3.3 Ubicación

La ubicación de los Honeypots es esencial para maximizar su efectividad, ya que debido a su carácter intrínsecamente pasivo una ubicación de difícil acceso eliminará gran parte de su atractivo para potenciales atacantes. Por otro lado, si su ubicación es demasiado artificial u obvia cualquier experimentado atacante la descubrirá y evitará todo contacto con ella.

Por otro lado debemos tener en cuenta que debe integrarse con el resto del sistema que tenemos ya implantado (servidores WWW, servidores de ficheros, DNS…) y asegurarnos que no interfiere con las otras medidas de seguridad que puedan ya existir en nuestra red (Firewalls, IDS…).

Teniendo en cuenta que los Honeypots pueden servir tanto para la detección de atacantes internos como externos, debemos tener siempre en cuenta la posibilidad de establecer Honeypots internos para la detección de atacantes o sistemas comprometidos en nuestra red (por ejemplo sistemas infectados con un gusano o virus).

La bibliografía consultada [Eve00][BP02][Lev03-2] establece tres puntos básicos candidatos a albergar los Honeypots según nuestras necesidades:

1. **Antes del firewall** (*Front of firewall*): Esta localización permite evitar el incremento del riesgo inherente a la instalación del Honeypot. Como este se encuentra fuera de la zona protegida por el firewall, puede ser atacado sin ningún tipo de peligro para el resto de nuestra red (ver figura 4-7).

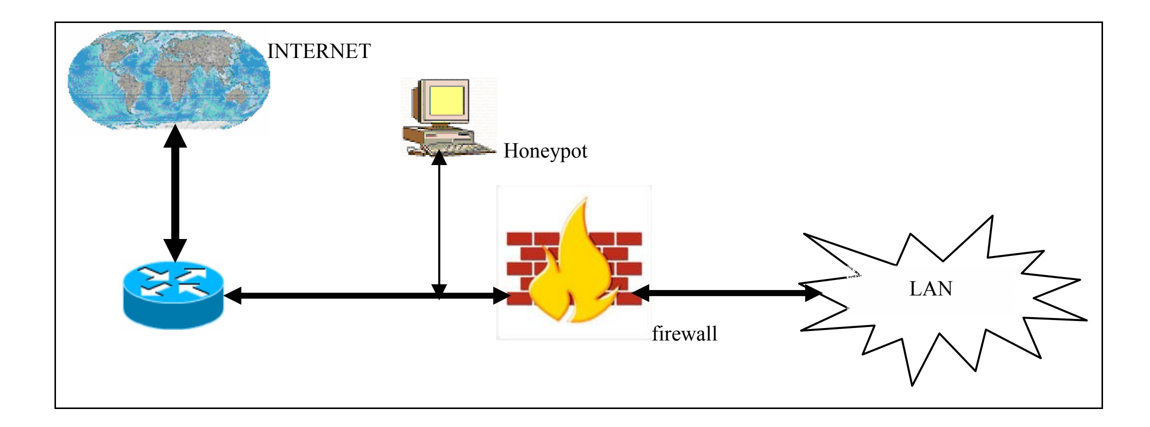


FIG 4-7: Ubicación del Honeypot antes del firewall.

Esta ubicación nos permite tener un acceso directo a los atacantes, puesto que el firewall ya se encarga de filtrar una parte del tráfico peligroso o no deseado, obteniendo trazas reales de su comportamiento y estadísticas muy fiables sobre la cantidad y calidad de ataques que puede recibir nuestra red.

Además con esta configuración evitaremos las alarmas de otros sistemas de seguridad de nuestra red (IDS) al recibir ataques en el Honeypot. Sin embargo, existe el peligro de generar mucho tráfico debido precisamente a la facilidad que ofrece el Honeypot para ser atacado.

Cualquier atacante externo será lo primero que encuentra y esto generará un gran consumo de ancho de banda y espacio en los ficheros de log. Por otro lado, esta ubicación nos evita la detección de atacantes internos.

2. **Detrás del firewall** (*Behind the firewall*): En esta posición, el Honeypot queda afectado por las reglas de filtrado del firewall. Por un lado tenemos que modificar las reglas para permitir algún tipo de acceso a nuestro Honeypot por posibles atacantes externos, y por el otro lado, al introducir un elemento potencialmente peligroso dentro de nuestra red podemos permitir a un atacante que gane acceso al Honeypot un paseo triunfal por nuestra red.

   La ubicación tras el firewall nos permite la detección de atacantes internos así como firewalls mal configurados, máquinas infectadas por gusanos o virus e incluso atacantes externos (ver figura 4-8).

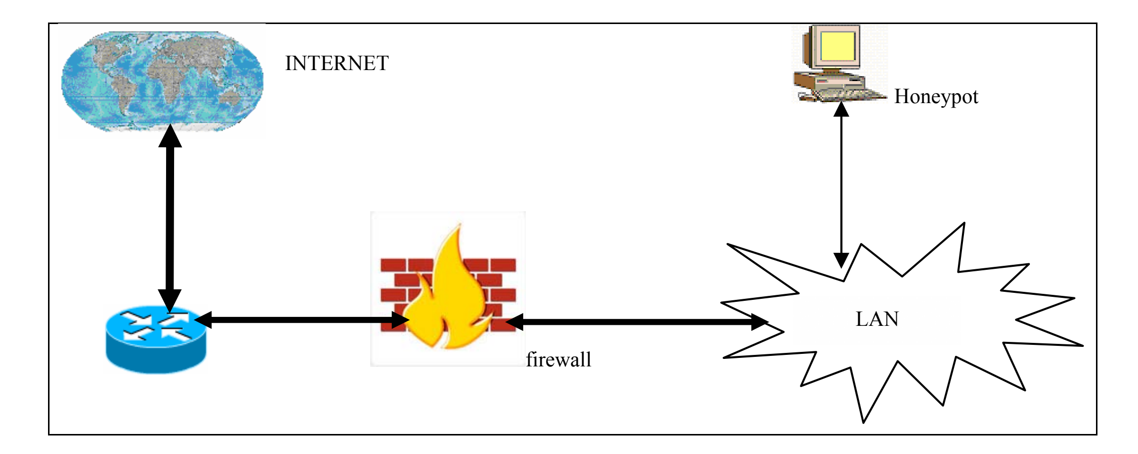


FIG 4-8: Ubicación del Honeypot tras el firewall.

Sin embargo las contrapartidas mas destacables son la gran cantidad de alertas de seguridad que nos generarán otros sistemas de seguridad de nuestra red

(Firewalls, IDS…) al recibir ataques el Honeypot y la necesidad de asegurar el resto de nuestra red contra el Honeypot mediante el uso de firewalls extras o sistemas de bloqueo de acceso, ya que si un atacante logra comprometer el sistema tendrá vía libre en su ataque a toda nuestra red.

Hay varias circunstancias que obligan a este tipo de arquitectura, como por ejemplo la detección de atacantes internos o la imposibilidad de utilizar una dirección IP externa para el Honeypot.

3. **En la zona desmilitarizada** (*into DMZ*[ℜ]): La ubicación en la zona desmilitarizada permite por un lado juntar en el mismo segmento a nuestros servidores de producción con el Honeypot y por el otro controlar el peligro que añade su uso, ya que tiene un firewall que lo aísla de resto de nuestra red local (ver figura 4-9).

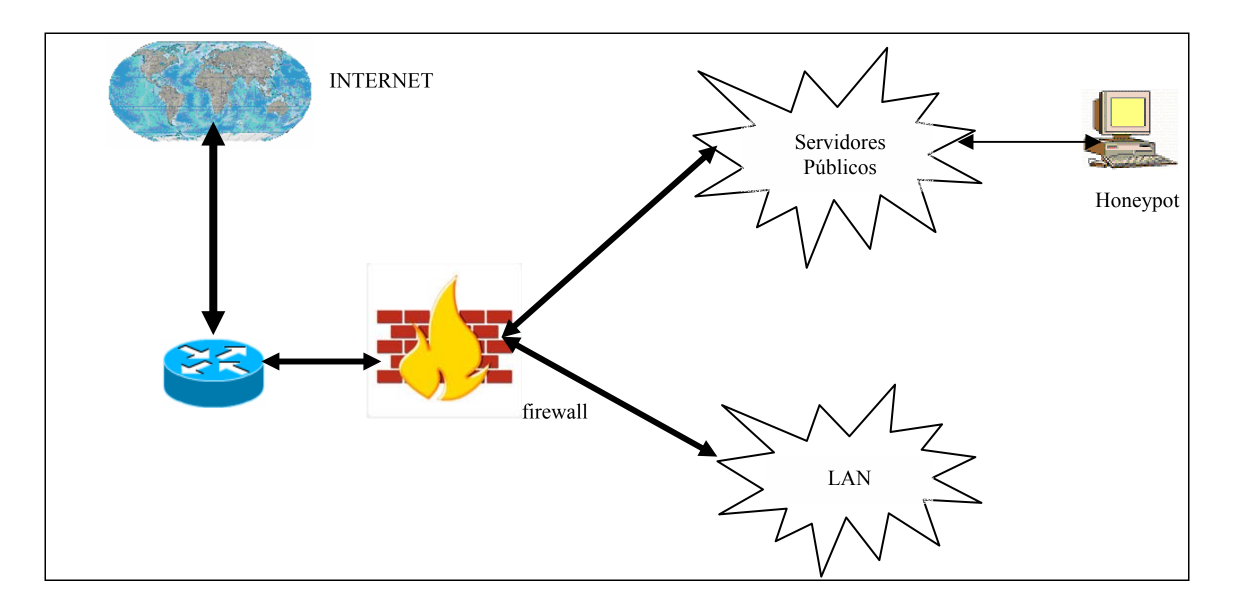


FIG 4-9: Ubicación del Honeypot en la zona desmilitarizada (DMZ).

Esta arquitectura nos permite tener la posibilidad de detectar ataques externos e internos con una simple reconfiguración de nuestro sistema de firewall puesto que se encuentra en la zona de acceso público.

[ℜ] Para ver una descripción sobre DMZ ver [WWW137].

Además eliminamos las alarmas de nuestros sistemas internos de seguridad y el peligro que supone para nuestra red al no estar en contacto directo con esta.

La detección de atacantes internos se va algo debilitada, puesto que al no compartir el mismo segmento de red que nuestra LAN un atacante local no accederá al Honeypot. Sin embargo, desde la red local si que es posible acceder al Honeypot, con lo que un atacante interno que intente atacar nuestros servidores públicos u otros sistemas externos (un gusano por ejemplo) muy probablemente acabe siendo detectado.

4.3.4 HoneyTokens

Cabe señalar que aunque históricamente se ha asociado erróneamente un Honeypot con un ordenador, esto no solo no es cierto, sino que la misma definición se desmarca de esta mala interpretación definiéndolo como un recurso de red, no como un ordenador o recurso físico concreto.

De la misma forma, podemos definir Honeytoken como "*un recurso digital de cualquier tipo (un documento Word, Excel, un fichero de música, un número de tarjeta de crédito…) destinado únicamente a interaccionar con posibles atacantes*" [Spi03-3].

Este concepto de "información falsa" en sí mismo no es nuevo[ℜ] [Sto00], sin embargo la denominación de Honeytoken nace en 2003 de la mano de Augusto Paes de Barros en una serie de discusiones mantenidas en las listas de correo de Honeypots [Spi03-3] y [WWW119][WWW130].

Un Honeytoken es un recurso que siempre que sea utilizado u accedido lo será de forma ilícita, y por tanto lo utilizará únicamente un atacante.

Obviamente, estos recursos deben cumplir dos requisitos fundamentales:

[ℜ] Por ejemplo muchas empresas cartográficas introducían calles inexistentes en los callejeros para detectar la venta de guías plagiadas.

1. No deben tener valor real ni afectar de ninguna manera que se comprometan o divulguen.

2. Deben ser similares a uno real. El objetivo es que el atacante se lo "crea" totalmente y confíe en su autenticidad como si fuera de verdad. Verosimilitud.

Estas características intrínsecas debido a su construcción, les confieren las mismas ventajas y características que los Honeypots [Spi03-3][WWW130]. Otra característica que les hace interesantes es lo simple que resulta utilizarlos. No necesitamos ordenadores, ni licencias de ningún tipo, un simple fichero creado en 3 minutos o un número de tarjeta inventado nos bastan para tener un perfecto Honeytoken.

Un ejemplo de funcionamiento de esta técnica, consiste en introducir un número de tarjeta de crédito falso (por ejemplo el **4356974837584710**) en nuestra base de datos. El número es único y cumple los dos requisitos anteriores, es plausible y nadie de forma natural accederá nunca a él.

Simplemente necesitamos activar nuestros sensores para la búsqueda de esta información viajando por la red (por ejemplo IDS o Firewall que examine los paquetes IP) y tendremos un acceso no autorizado.

Un eventual atacante tratará de conseguir cualquier (o incluso toda) información posible, con lo que no podrá distinguir si es un Honeytoken o es un valor correcto. Para entornos de comunicación cifrados, simplemente se han de desarrollar herramientas a nivel del sistema operativo que interactúen con las llamadas del *kernel* de lectura (read()).

Cabe señalar que la detección de un Honeytoken circulando por nuestra red, no implica necesariamente que esta haya sido comprometida, sin embargo sí prueba la existencia de un comportamiento inadecuado e identifica a las partes implicadas. Una vez detectado, otros medios complementarios (IDS, ficheros de logs…) deben probar la culpabilidad real de las partes.

4.3.5 WI-FI Honeypots

Inicialmente las redes de comunicaciones fueron expandiéndose lentamente por centros militares, universitarios y centros de investigación hasta el estallido de Internet. A partir de este momento la obsesión por estar conectados ha llevado a la instalación de millones de metros de cables que cubren todo tipo de edificios y superficies.

Sin embargo, el ansia/dependencia de Internet continúa alimentando el deseo de estar conectados en cualquier sitio y a cualquier hora. Bares, librerías e incluso campus universitarios están iniciando un nuevo paso en las comunicaciones, las redes inalámbricas (*wireless networks*).

Obviamente, no podemos descartar el papel de catalizador que la gran expansión de los teléfonos móviles ha tenido en el desarrollo de estas redes sin hilos. Actualmente ya existen varios estándares de IEEE que regulan estas comunicaciones como el 802.11b y el 802.11g [WWW141].

Como ocurre con cualquier red de comunicaciones, los accesos no autorizados o los ataques perpetrados por hackers están a la orden del día. Sin embargo, el peligro de este tipo de redes se dispara debido a tres factores diferenciales:

1. Los estándares y especificaciones para este tipo de redes son "nuevos" y poco probados. Como siempre que se realiza una cosa por primera vez, la existencia de indefiniciones, malas interpretaciones o fallos de conceptos es bastante probable. Estos aspectos difusos son los más utilizados por los atacantes para obtener acceso a este tipo de redes.

2. Cualquiera con un simple portátil (o incluso con un PDA) puede intentar acceder de forma fraudulenta a nuestra red inalámbrica. A diferencia de otros tipos de redes, no necesita estar físicamente conectado o estar confinado en un área concreta. Los ataques pueden venir de cualquier sitio.

3. La mayor ventaja de las redes sin hilos es la movilidad que proporcionan. Este aspecto también puede jugar en nuestra contra, ya que el atacante puede ir cambiando de ubicación mientras realiza su ataque.

La utilización de Honeypots en las redes inalámbricas [Pou02] es una de las últimas aplicaciones que se están probando con éxito en Estados Unidos. Actualmente el programa **WISE** (*Wireless Information Security Experiment*) [WWW143] lidera la investigación en este campo.

4.3.6 Repercusiones legales

Hasta ahora, en los diferentes capítulos de este trabajo se han presentado diferentes herramientas y/o sistemas que permiten de una forma u otra aumentar la seguridad de nuestra red.

En este punto se comentarán brevemente (puesto que va más allá de las pretensiones de este trabajo) las implicaciones legales que pueden acarrear el uso de Honeypots. Cabe destacar que las siguientes líneas hacen referencia al código de leyes norteamericano, y como tal NO es aplicable a España o la CEE. Sin embargo, debido al peso específico de los Estados Unidos debe de ser tenido en cuenta siempre[ℜ]…

La ley norteamericana (al igual que la europea) protege la inviolabilidad de las comunicaciones personales (*interception of communications*) de una forma muy estricta. El punto de discusión se basa en que dependiendo del tipo de Honeypot que utilicemos, podemos violar esta ley al recoger una serie de información sobre nuestro atacante que la ley protege explícitamente [Pou03].

Tal y como se ha explicado anteriormente, los Honeypots de producción tienen un objetivo mucho mas concreto (proteger nuestra red) que los Honeypots de investigación

[ℜ] Los EEUU suelen ser los primeros en regular todo lo referente a la seguridad informática, y en especial a cualquier cosa que afecte Internet. Muchas leyes estatales y/o europeas se basan en las americanas, cosa que incrementa su interés para cualquier profesional.

(cuyo interés se centra en conseguir tanta información de los atacantes como sea posible para comprender/estudiar su comportamiento y técnicas).

Dependiendo del grado de configuración de nuestro Honeypot, podemos "espiar" los movimientos y comunicaciones que realice nuestro atacante dentro de nuestra Honeypot (hacia nuestra red o hacia el propio Honeypot) y desde nuestra Honeypot hacia terceros.

Precisamente en este "espionaje" de las comunicaciones que realice el atacante desde nuestra red hacia terceros sitios viene el problema. Estamos espiando una comunicación entre terceros sin ningún tipo de autorización para ello.

El surrealismo aumenta cuando este supuesto permite al atacante denunciarnos por invadir su intimidad y ganar el caso [Pou03][ℜ].

De todo esto, podemos observar que antes que cualquier otra cosa debemos responder a la siguiente pregunta [Pou03][Spi03-4]: **« ¿Cuál es el objetivo de mi Honeypot? »**

De que la respuesta sea únicamente "proteger mi red" a que sea "conocer la identidad de los atacantes" o "recoger información de posibles ataques o técnicas nuevas" puede depender ganar o perder un juicio.

Lance Spitzner [Spi03-4] desglosa las posibles responsabilidades legales derivadas del uso de Honeypots en tres cuestiones básicas (obviamente siguiendo el sistema de leyes norteamericano):

1. **Trampa** (*Entrapment*): Es el proceso realizado por los cuerpos policiales (*law enforcement*) de "inducir" a alguien a cometer un acto punible con el objetivo de iniciar la acción judicial pertinente [Gar99]. En este caso del Honeypot, aunque es un elemento pasivo creado por nosotros para ser atacado (sin que seamos parte de los cuerpos policiales) si no deseamos perseguir judicialmente esta intrusión en el Honeypot, no realizamos ninguna trampa. El objetivo del Honeypot es recibir los ataques, no recoger información para demandar a los atacantes del Honeypot.

[ℜ] Obviamente en EEUU. Pero teniendo en cuenta que este país no duda en ampliar su jurisdicción a terceros países (hay varios precedentes como casos de patentes de marcas) debemos ser precavidos.

2. **Privacidad** (*Privacy*): Que el Honeypot recoge información es innegable. Sin embargo, la información recogida puede dividirse en información transaccional (*transactional*) e información de contenido (*content*).

   La información transaccional (meta-información) no hace referencia a la información en sí, sino a aspectos de esta como la dirección IP, la fecha y hora, valores de las cabeceras de los paquetes IP…

   La información de contenido es propiamente la comunicación que realiza en atacante con terceros. Precisamente este es el objetivo del debate (y también el de los Honeypots de investigación). La intercepción de una comunicación privada es la piedra angular que puede permitir a un atacante demandarnos ante un juzgado y probablemente ganar. En cualquier caso, todos los autores están de acuerdo que se deberían incluir mensajes de advertencia y renuncia (*disclaimer*) como el indicado a continuación en todos los Honeypots. Sin embargo esto no exime del problema, ya que el hecho de que pongamos un aviso no significa que un eventual atacante lo vea o lea…

```
############################################################
#                 READ BEFORE CONTINUING:                  #
#                                                          #
# This system is for the use of authorized users only.     #
# By using this computer you are consenting to having      #
# all of your activity on this system monitored and        #
# disclosed to others.                                     #
############################################################
```

3. **Responsabilidad** (*Liability*): Este aspecto hace referencia a las posibles demandas que podemos recibir en el caso de que un atacante utilice nuestro Honeypot como plataforma de lanzamiento de ataques.

   Las demandas se basarían en que nosotros no hemos realizado unos mínimos esfuerzos de seguridad en nuestra red, sino que al contrario, facilitamos el acceso a nuestros recursos para que sean utilizados en todo tipo de ataques. ¿Qué grado de culpabilidad tenemos cuando un atacante usas nuestros recursos para atacar a otros? ¿Es punible la incompetencia en materia de seguridad informática?

4.4 Honeynets

Históricamente, una vez definido el concepto de Honeypot y realizadas las primeras pruebas con éxito, se propuso la extensión de este concepto.

Podemos definir una Honeynet como "*un tipo concreto de Honeypot. Específicamente es un Honeypot altamente interactivo diseñado para la investigación y la obtención de información sobre atacantes. Una Honeynet es una arquitectura, no un producto concreto o un software determinado.*" [Hon03] [Pro03].

El nuevo enfoque consiste no en falsear datos o engañar a un posible atacante (como suelen hacer algunos Honeypot) sino que el objetivo principal es recoger información "real" de cómo actúan los atacantes en un entorno de verdad.

Para conseguir este entorno real (con sistemas reales, no con simples emulaciones de servicios) y altamente interactivo, se dispone una configuración de red típica con todos sus elementos (ver figura 4-10).

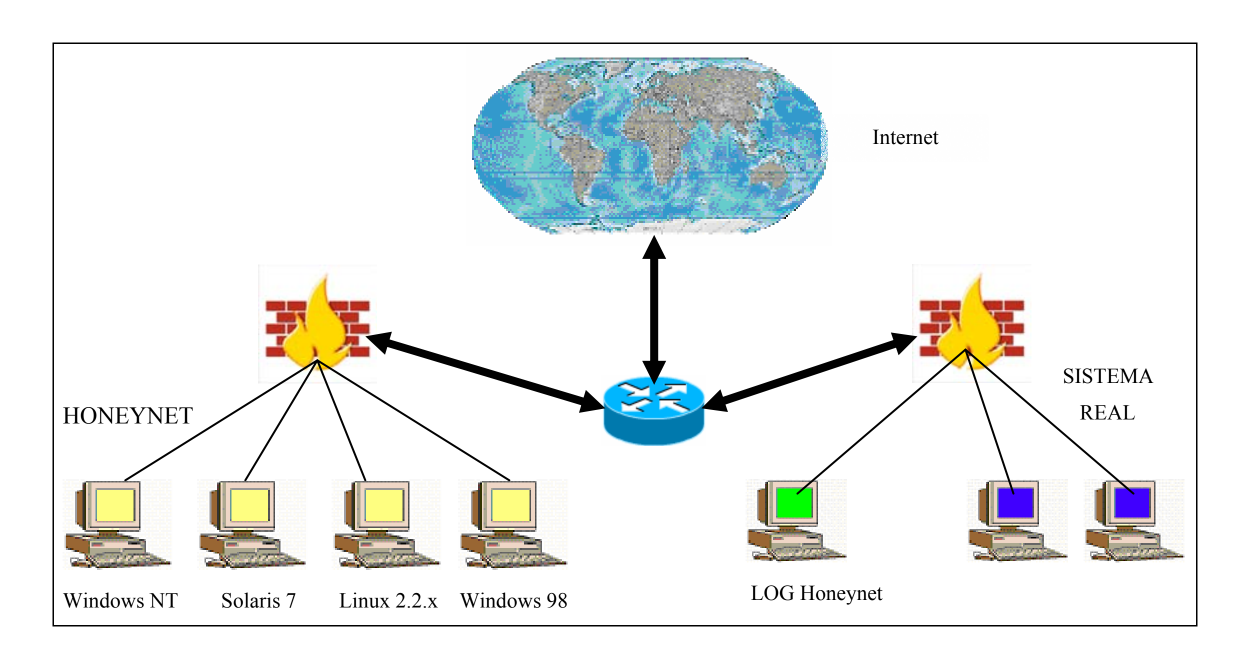


FIG 4-10: Arquitectura típica de una Honeynet.

Obviamente, esta red ha sido diseñada para ser comprometida, por lo que debe estar separada de forma segura y controlada de la de producción.

Por otro lado, como nuestro objetivo es el de hacer creer al atacante que está ante una red "real", debemos añadir los distintos elementos que conforman una arquitectura "normal" en cualquier red (distintas máquinas, distintos sistemas operativos…)

Una Honeynet presenta dos requerimientos básicos para ser realmente útil y que nos permita la extracción de información valiosa [Hon03][Hon03-2]:

1. **Control del flujo de datos** (*Data control*): Siempre que interactuamos con un atacante, el peligro aumenta exponencialmente. Todo y que el objetivo de la Honeynet es el de ser comprometida y atacada, debemos mantener siempre un control del flujo de datos para evitar que el atacante la utilice contra terceros o contra nuestra propia red.

   Si bien es cierto que cuanta más interacción permitamos con el exterior más datos reales podremos obtener del atacante, debemos evaluar los riesgos que conlleva. Análogamente, una Honeynet que no permita ningún tipo de actividad con el exterior dejará de ser atractiva para un atacante y perderá toda su utilidad. Como siempre, la búsqueda de un equilibrio nos debe guiar en este aspecto.

2. **Captura de datos** *(Data capture*): La captura de todos los movimientos y acciones que realice el atacante en nuestra Honeynet nos revelará sus técnicas y motivaciones. Si bien es esencial que el nivel de vigilancia y captura sea alto, si este es excesivo o detectado por el atacante dejará de ser efectivo.

   Obviamente, la captura de datos debe hacerse sigilosamente y sin despertar ningún tipo de sospecha, por lo que debe planificarse cuidadosamente.

El lugar dónde se almacena esta información debería encontrarse fuera de la Honeynet, ya que si realmente compromete un sistema puede encontrarla o incluso peor, falsearla o borrarla. Esto eliminaría cualquier utilidad a la Honeynet.

4.4.1 Valor añadido

Tradicionalmente, la mayoría de los sistemas de seguridad han sido siempre de carácter defensivo. IDS, Firewalls y demás soluciones se basan en la defensa de nuestros sistemas, y cuando un ataque o vulnerabilidad es detectado mirar de arreglar el problema tan rápido como sea posible.

Estas aproximaciones siempre van siempre un paso por detrás de los atacantes, ya que nuestra reacción depende directamente de los ataques sufridos y detectados. Si no recibimos ningún ataque o no descubrimos alguna vulnerabilidad, nuestros sistemas de defensa permanecerán "iguales". No hay mejora intrínseca o proactividad propia de los sistemas de ningún tipo.

Las Honeynets [Hon03-2] miran de cambiar esta actitud mediante el estudio de los ataques y atacantes. Obtener nuevos patrones de comportamiento y nuevos métodos de ataque con el objetivo de prevenirlos en los sistemas reales.

Sin Honeynets, cada vez que se produzca un ataque "nuevo" y exitoso a un sistema real existente, este dejará de dar servicio y se verá comprometido. Con las Honeynets, un ataque exitoso o nuevo no tiene porqué afectar a ningún sistema real.

Además perderá el factor sorpresa, ya que habremos obtenidos datos precisos de su ataque en el estudio de los logs, cosa que permitirán contrarrestarlo de una manera más eficiente.

Al igual que los Honeypots, la cantidad y calidad de información producida es muy importante, ya que cualquier actividad existente es sospechosa.

4.4.2 GEN I

Tal y como sucede en cualquier programa (software) o especificación formal (RFC por ejemplo), con el tiempo se van perfilando nuevas mejoras y ampliaciones de funcionalidades. Las Honeynets no son ninguna excepción a esta regla, y en el momento de escribir este capítulo, existen dos generaciones diferenciadas de Honeynets [Hon03].

Las Honeynets de primera generación (**GEN I**) se caracterizan por implementar únicamente los mecanismos básicos de control de flujo de datos (*Data control*) y captura de datos *(Data capture*). La arquitectura utilizada presenta una subdivisión en tres subredes (la Honeynet, la red de producción e Internet) separadas perfectamente por un firewall (ver figura 4-11).

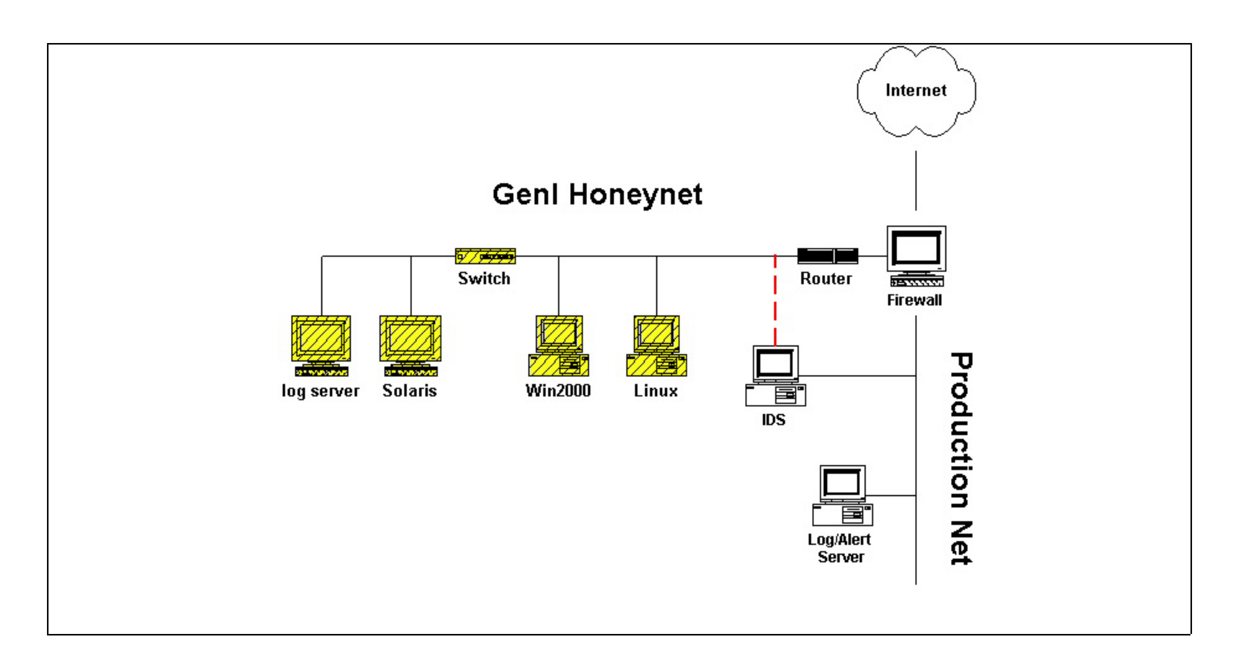


FIG 4-11: Arquitectura típica de una Honeynet de primera generación (GEN I).

En esta arquitectura, el control de todas las redes se realiza mediante un firewall, ya que cualquier paquete de entrada o salida debe pasa obligatoriamente por él. El router también se utiliza como medio de control y filtrado, pero únicamente como soporte al firewall.

El firewall está configurado para llevar la cuenta del número de conexiones que se establecen tanto hacia dentro como hacia fuera. De esta forma, cuando el número de conexiones desde nuestra Honeynet hacia el exterior supera un límite preestablecido, estas quedan anuladas. El límite de conexiones depende del lo atractiva que deseamos hacer nuestra red a un eventual atacante, se recomiendan valores entorno a las diez conexiones por hora [Hon03].

Una red que permita cualquier tipo y número de conexiones al exterior es un gran atractivo para cualquier atacante. Sin embargo, también es un riesgo potencial demasiado grande, ya que puede ser utilizada como trampolín de ataque a otros sistemas externos. Por otro lado, una que no permita conexiones al exterior carece de interés y por lo tanto eliminará la obtención de datos "reales" (*in the wild*) que es nuestro principal objetivo.

En esta configuración, el firewall se coloca expresamente por delante del router. Los motivos de esta configuración son por un lado esconder/disimular el firewall a los atacantes y por otro dotar de medidas extras de seguridad a nuestra Honeynet para que no sea utilizada para ataques a terceros, ya que el router también realizará parte del filtrado de las comunicaciones de entrada y salida. Nuestro control de flujo de datos (*Data control*) no debe depender únicamente de un solo punto de control[ℜ].

Cuando un atacante tome el control de nuestra Honeynet, encontrará que puede salir a Internet directamente, ya que pasa por un router real de producción, asumiendo que el filtrado de paquetes que exista (nuestro firewall) será un filtrado realizado por nuestro ISP y no por nosotros. De esta forma reforzamos su opinión de que somos una red "desvalida".

La captura de datos (*Data capture*) se inicia en el propio firewall, ya que es un dispositivo lo suficientemente potente y colocado estratégicamente para recibir todos los paquetes de entrada y salida. Como sabemos que cualquier tipo de actividad hacia o desde nuestra Honeynet es potencialmente peligroso, debemos capturar todo este tráfico en el firewall.

[ℜ] Esto nos permite aplicar técnicas de ingres/egress filtering así como filtros contra posibles ataques DOS/DDOS

La captura de puertas traseras (*back doors)* en nuestros sistemas es otra de las posibilidades que nos brinda este tipo de arquitectura, ya que una correlación de peticiones a puertos "no standards" revelará rápidamente su existencia.

La captura de todo el tráfico existente en nuestra Honeynet es realizado por el sistema de detección de intrusos (IDS). En la arquitectura de GEN I podemos observar como el IDS está conectado a nuestra Honeynet de forma que por un lado registra la actividad existente en la red y por otro actúa como un detector de firmas (*signatures*)

Finalmente, toda esta información se va copiando diariamente en un ordenador "seguro" para poder analizarla o consultarla como un histórico. Esto nos asegura que en caso de que el atacante encuentre la información de sus movimientos y decida destruirla no afecte excesivamente al sistema.

Obviamente, la existencia de un ordenador-almacén puede ser fácilmente descubierta por un atacante. Sin embargo esta arquitectura no trata de esconderlo, ya que lo más que puede hacer el atacante es desactivar el "*log daemon*" de la máquina atacada, comportamiento ya muy habitual en muchos atacantes.

En caso de que el atacante decidiera atacar directamente a nuestro sistema de almacenamiento "seguro", debería enfrentarse a un sistema mucho mas seguro. Incluso en el caso de que lograra entrar y eliminar todos los logs del sistema no sería catastrófico, puesto que el sistema IDS registra toda la actividad de la red y podríamos ver al sistema IDS como un segundo log del sistema.

4.4.3 Gen II

Las Honeynets de segunda generación (**GEN II)** se caracterizan por combinar todos los requisitos de la primera generación en un solo dispositivo (ver figura 4-12). De esta forma tanto el control de flujo de datos (*Data control*) como la captura (*Data capture*) y la recolección de datos quedan agrupadas en una misma entidad [Hon03-3].

Esta entidad es un dispositivo de capa 2 (*Layer 2 gateway*) con capacidades de transmisión de paquetes tipo puente (*bridge*). Esta capacidad de puente hace referencia a la transparencia con que actúa este dispositivo de red, simplemente se encarga de enviar por un extremo lo mismo que recibe por el otro, lo que le convierte en un dispositivo “invisible”.

El hecho de presentarse como un dispositivo de capa 2 hace que este dispositivo carezca de pila IP, por lo que no generará ningún tipo de tráfico que el atacante pueda observar.

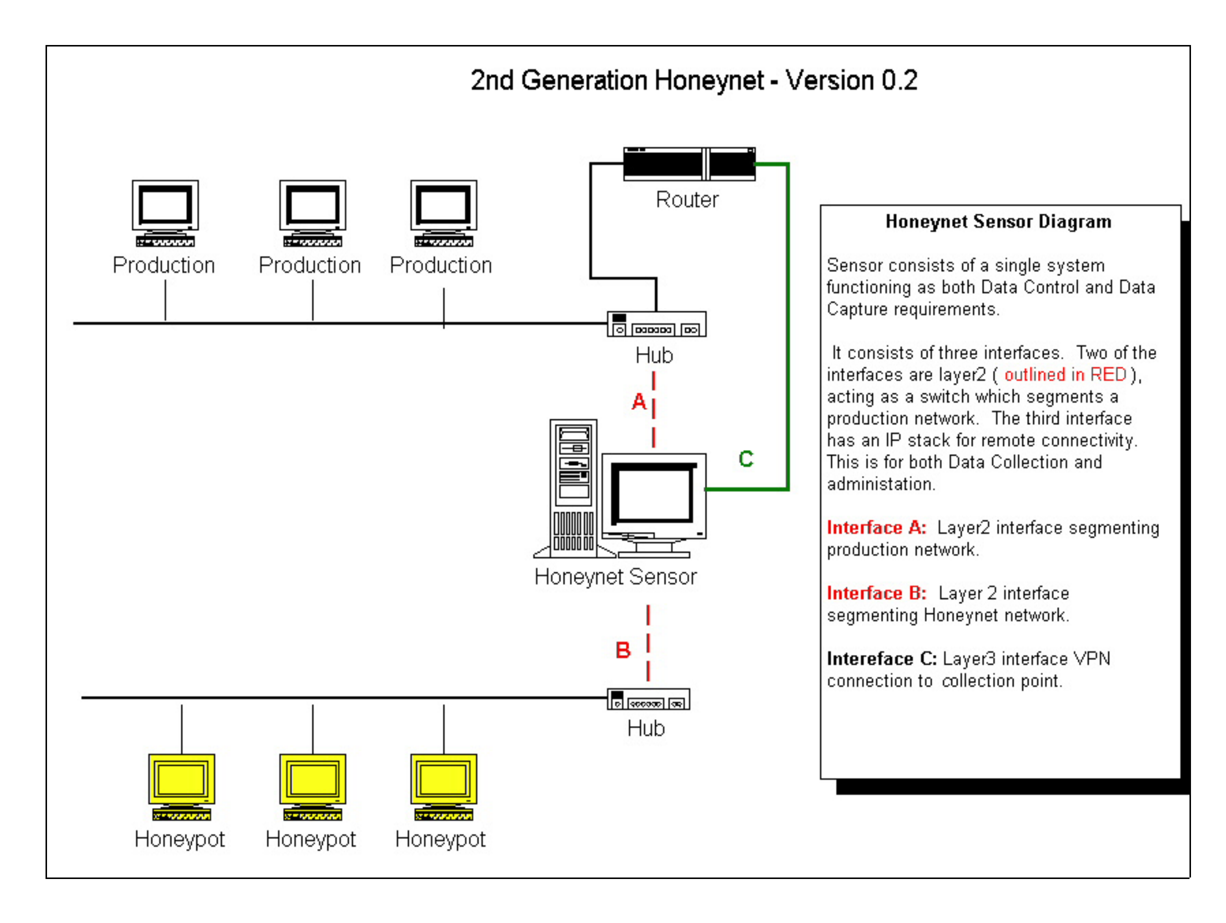


FIG 4-12: Arquitectura típica de una Honeynet de segunda generación (GEN II).

En alguna bibliografía [Hon03-3] tanto para GEN I como para GEN II, se denomina al dispositivo que controla todo el acceso de entrada y salida a nuestra Honeynet **Honeywall**. Esta nomenclatura viene dada porque conceptualmente este dispositivo es el muro que separa nuestra Honeynet del resto de las redes.

Esta unión de capacidades en una única entidad compacta facilita la instalación y administración de la Honeynet. Debido a su actuación como dispositivo transparente, la detección por parte de un atacante se dificulta enormemente.

Pese a que en el esquema parezca que hay dos redes (la de producción y la Honeynet) realmente sólo hay una. El dispositivo es de capa 2, y por lo tanto NO tiene dirección IP en los interfaces A y B, lo que "virtualmente" significa que a nivel de IP no existe. El dispositivo C si que tiene asignada una dirección IP y servirá como canal seguro (VPN) para el control, administración del Honeywall y para el movimiento de los ficheros de logs hacia un repositorio "seguro".

La primera mejora que introduce esta arquitectura hace referencia al control del flujo de datos. En la GEN I realizábamos un control del total de conexiones desde nuestra Honeynet hacia el exterior. De forma que si no se alcanzaba un umbral definido, las conexiones hacia el exterior eran "permitidas".

Con este paradigma de GEN II, obtenemos un control total de todas y cada una de las conexiones que pueda realizar, independientemente de si su número es mayor, menor o igual al límite existente en las Honeynet de GEN I.

La segunda mejora que obtenemos hace referencia directa a la forma en que podemos responder ante las actividades no autorizadas desde nuestra Honeynet. En lugar bloquear simplemente cualquier tipo de acceso no permitido, podemos modificar y dosificar la actividad del atacante. De esta forma, el ataque como tal sale de la Honeynet (cosa que puede ver y comprobar el atacante) sin embargo no es efectivo debido a las modificaciones que podemos realizar en tiempo real sobre el paquete de datos (cosa que muy difícilmente puede detectar el atacante).

Esta capacidad añade credibilidad ante el atacante, que verá como sus paquetes "salen" hacia sus nuevas víctimas y retornan sin ningún tipo de filtro o control. Sin embargo, estos paquetes son modificados en tiempo real de forma que si utiliza una vulnerabilidad del servicio FTP, sea detectada por nuestro sistema y modifique el paquete de forma que sea inocuo.

Los paquetes entran y salen "libremente", sin embargo su contenido no. Esta nueva característica añade las capacidades de un sistema NIPS (*Network Intrusion prevention System*) a las capacidades de bloqueo ya existentes en la GEN I.

Este sistema es tan flexible que no sólo permite modificar los paquetes que salen hacia fuera, sino que incluso nos permite modificar la respuesta que el atacante obtiene por parte del sistema externo que quiere atacar. Podemos por ejemplo devolverle paquetes TCP-RST (reset) simulando que el sistema atacado rechaza sus peticiones de control.

La captura de la actividad que realiza el atacante en cualquiera de los sistemas incluidos en nuestra Honeynet ha variado substancialmente en los últimos años. Las capturas directas del teclado o las capturas de paquetes de red utilizando *snifers* son inefectivas ante el uso de SSH o cualquier otro sistema de comunicación cifrado.

La herramienta SEBEK2 [WWW148] permite la captura de este tipo de información en espacio de kernel. No entraremos detalles puesto que su funcionamiento se sale de los propósitos de este trabajo. Para ampliar este punto consultar la bibliografía proporcionada.

4.4.4 Honeynets virtuales

Uno de los problemas mas graves a los que se enfrenta cualquier administrador de redes y por tanto el grupo de seguridad, es el de la disponibilidad de recursos. Los esquemas presentados (GEN I y II) cada vez demandan mas máquinas y recursos, lo que puede llevar a la paradoja de tener dos servidores de producción y cuatro ordenadores en una Honeynet.

Obviamente, este punto es algo exagerado, sin embargo si que puede pasar que mucha gente en pequeñas empresas descarte los Honeypots/Honeynets simplemente porque no tiene los recursos necesarios o porque estos son iguales o superiores a las máquinas de producción.

El concepto de **Honeynet virtual** se puede definir como "*la solución que permite implantar el esquema de Honeynet utilizando un único ordenador. El adjetivo virtual viene dado porque todos los sistemas operativos que existen en la Honeynet aparentemente tienen su propia máquina, aunque realmente se ejecutan en el mismo hardware*." [Hon03-4].

Las Honeypots virtuales pueden clasificarse en dos grupos:

- **Honeynets autocontenidas** (*Self-contained virtual Honeynet*): En este tipo de Honeypots virtuales todo el software se ejecuta en una misma máquina (ver figura 4-13). Usualmente una Honeynet se compone de un firewall o Honeywall encargado de controlar el acceso así como de registrar los diferentes logs y de uno o varios Honeypots.

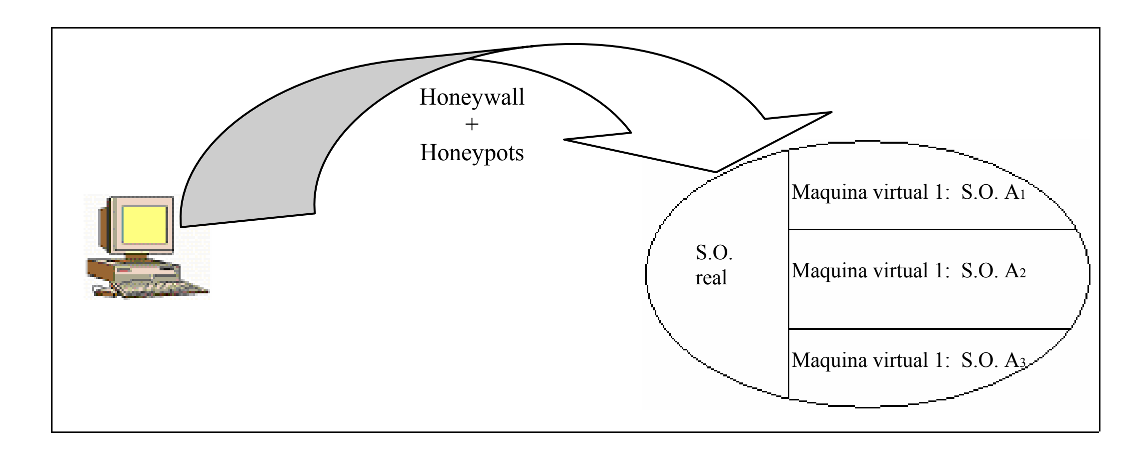


FIG 4-13: Honeynet autocontenida (self-contained).

De esta forma primero instalamos el sistema operativo anfitrión (real) y después el software de emulación virtual[ℜ]. A continuación realizamos para cada software de emulación la instalación del sistema operativo correspondiente y finalmente realizamos la configuración de cada Honeynet.

Sus principales ventajas son el reducido número de recursos necesarios y la facilidad de instalación.

[ℜ] Existe mucho software de emulación disponible, los más populares actualmente son el VMWARE y el WINE.

Por el contrario, tenemos un único punto de servicio (si la máquina se estropea o el atacante se hace con el control, quedamos sin ningún tipo de protección).

- **Honeypots híbridas** (Hybrid virtual Honeynet): En este caso se produce una división entre el Honeywall (punto de entrada, control y recolección de información de la Honeynet) y los Honeypots existentes en la Honeynet (ver figura 4-14).

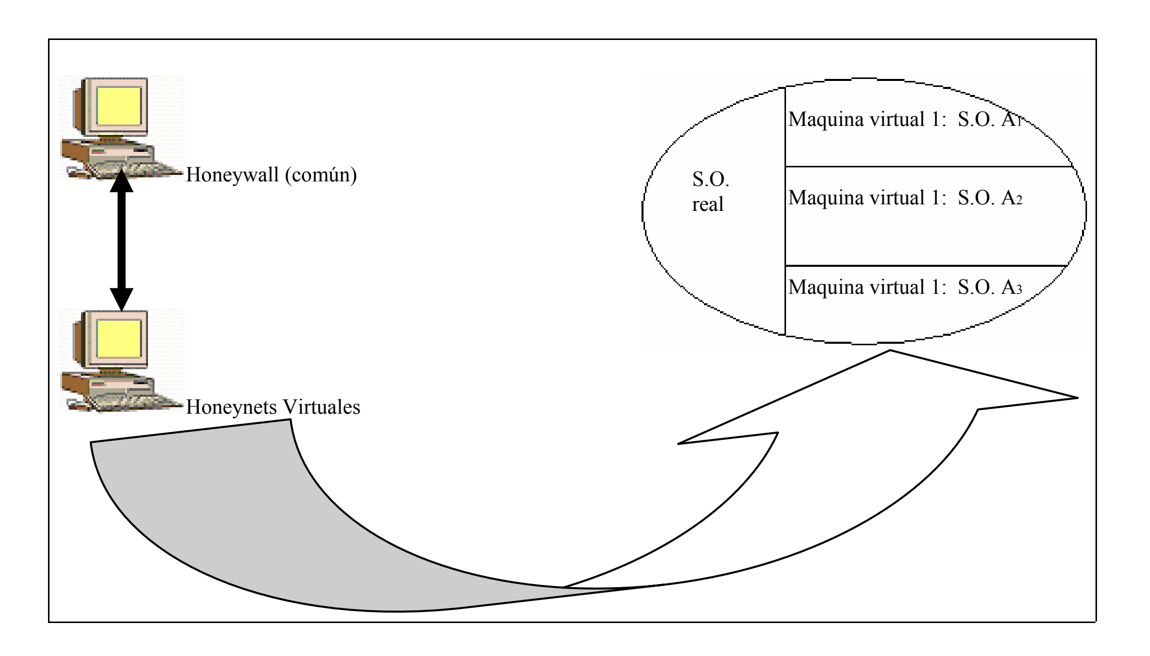


FIG 4-14: Honeynet híbridas (Hybrid).

En este esquema el peligro queda reducido al acceso del atacante al resto de Honeynets. Necesitamos un ordenador más y la configuración es algo más compleja. Por el contrario, ganamos en flexibilidad y seguridad ya que tenemos el control de acceso y los logs fuera de la Honeynet virtual.

Podemos ver que ambos esquemas tienen ventajas e inconvenientes, sin embargo las contrapartidas comunes al uso de esta implantación de las Honeynets virtuales son principalmente cuatro:

1. El hardware de que disponemos limita la cantidad de sistemas operativos a simular. Si tenemos un PC (arquitectura ix86) nunca podremos emular Solaris de SPARC, un AIX de RS6000 o IRIS de Silicon Graphics.

2. La potencia de la máquina empleada marca la cantidad de Honeynets que podemos realizar. Si tenemos un PC 486 con 8MB de RAM no tiene sentido intentar emular varias Honeynets con decenas de ordenadores distintos.

3. Si el atacante toma el control de la máquina, podría obtener el control de todos los sistemas virtuales

4. Las técnicas de *fingerprinting* (obtención del tipo y versión del sistema operativo mediante el envío de paquetes IP específicamente construidos) podrían revelar la naturaleza real de nuestro sistema operativo "base". Incluso aunque en principio esta técnica no funcionara, si el atacante toma el control de una de las Honeynets probablemente pudiera averiguar que se encuentra en un sistema simulado, lo que falsearía su comportamiento.

4.4.5 Honeynets distribuidas

El siguiente paso que se está realizando con las Honeynets es la aplicación del viejo principio de "**la unión hace la fuerza**". El escenario con el que nos encontramos puede ser el de una gran organización internacional con múltiples redes en múltiples países o varios equipos de expertos en seguridad diseminados por todo el planeta (ver figura 4-15) que desean compartir la información generada por sus Honeynets [Pro03].

Obviamente, la posibilidad de centralizar (o al menos comunicar) las distintas Honeynets para la recolección de información es básico puesto que nos permitirá la correlación de resultados así como la comunicación de nuevos descubrimientos de forma más rápida y eficiente.

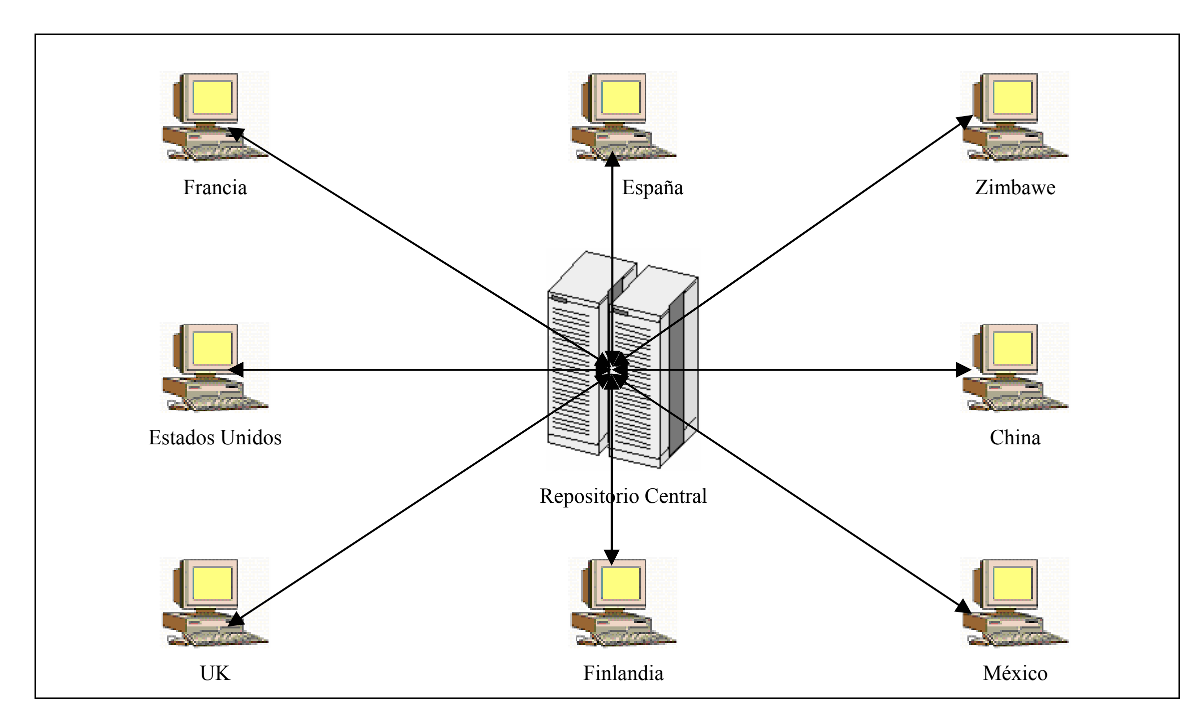


FIG 4-15: Arquitectura distribuida de Honeynet.

Una vez que ya tenemos definidas unas arquitecturas estables (GEN II) y lo suficientemente flexibles como para permitir la interconexión de los sistemas, el siguiente paso es plantearse cómo hacerlo.

El modelo escogido es mediante la creación de túneles privados virtuales cifrados con **IPSec** (*IPSec VPN*). De esta forma, una vez al día los diferentes Honeywalls se conectarán por el interface C (ver figura 4-12) de forma segura al repositorio central para depositar los ficheros de log del día anterior.

4.5 RESUMEN

En este capítulo hemos examinado el cambio de escenario de los ataques informáticos que ha producido el avance tecnológico de las redes de ordenadores. Se ha introducido la necesidad de nuevos modelos de seguridad para cubrir este nuevo espacio concluyendo en la presentación del concepto de Honeypot.

Los **Honeypots** (tarro de miel) son elementos pasivos diseñados para ser atacados que nos permiten examinar el comportamiento de los atacantes en un medio interactivo.

Se han comentado sus tipos (de producción o investigación), ubicaciones posibles (antes del firewall, tras el firewall o en la DMZ), puntos fuertes (uso mínimo de recursos, facilidad de integración en las redes, detección, análisis) y débiles (introducen un riesgo potencial, no disuaden a los atacantes y no arregla fallos de seguridad ya existentes en nuestra red) así como las repercusiones legales de su uso según la legislación de Estados Unidos.

También se ha introducido el concepto de **Honeytoken** (recurso digital diseñado para ser comprometido) así como sus características principales (económicas y fáciles de generar) y valor añadido (detección de actividad ilícita).

Definimos **Honeynet** como un conjunto de Honeypots destinada a recoger información sobre potenciales atacantes. Describimos los componentes de su arquitectura interna (control de flujo de datos y captura de datos) así como las características de las dos generaciones (GEN I y GEN II) existentes en la actualidad.

Finalmente hemos introducido el concepto de **Honeynet virtual** (solución que permite implantar el esquema Honeynet con un único ordenador) como respuesta a las necesidades de economía de recursos así como sus diferentes tipos (autocontenidas e híbridas) y características.

PARTE II:

PARTE EXPERIMENTAL

CAPITULO 5

Análisis de un sistema conectado a Internet

En este último capítulo de nuestro trabajo dedicado a la seguridad en las redes IP presentaremos la parte práctica o experimental realizada.

Inicialmente realizaremos una presentación de los objetivos que se persiguen en este experimento (deseamos monitorizar una conexión permanente a Internet durante una semana) así como los requisitos necesarios. A continuación analizaremos las distintas posibilidades que podemos utilizar teniendo en cuenta los requerimientos necesarios para ir centrándonos en la arquitectura propuesta.

Posteriormente realizaremos un análisis detallado de todo el tráfico de red obtenido durante la semana, comentando los distintos resultados obtenidos y desglosándolos por día. También analizaremos los resultados globales desde una perspectiva semanal centrándonos en los aspectos más relevantes.

Finalmente describiremos las conclusiones de este experimento que se obtendrán partir del análisis de los informes anteriores y del tráfico registrado durante la semana.

5.1 Objetivos

En los capítulos anteriores de este trabajo hemos ido comentando diferentes sistemas y tecnologías que permitían un incremento de la seguridad en nuestras redes de ordenadores (sistemas de detección de intrusos o IDS, Honeypots…). Cada pocos días vemos noticias referentes a nuevas amenazas en Internet (gusanos, virus, *hackers*…) sin embargo ¿realmente tanta falta de seguridad existe en Internet? ¿Tantas son las fuentes de peligro que justifican cada vez más inversión y mejores sistemas de seguridad?

En la actualidad existen cientos de informes que justifican todas y cada una de las miles de soluciones informáticas de seguridad existentes. Sin embargo, prácticamente siempre estos mismos informes son realizados (o peor aún, financiados) por las mismas empresas que venden esos productos.

La bibliografía y las notas de prensa [Far96][Wei00][Bor03][WWW159][WWW160] [WWW161][WWW163][WWW164] están llenas de referencias y datos que indican siempre un aumento espectacular de la inseguridad en Internet. No obstante, cuando deseamos investigar más a fondo los datos o conclusiones que presenta un informe y planteamos preguntas del tipo ¿dónde se ha realizado? ¿quién exactamente lo ha realizado? ¿cómo se ha realizado? ¿qué criterios de verificación de los datos se han seguido? Nos encontramos sin ninguna respuesta satisfactoria.

En este capítulo realizaremos un estudio consistente en la monitorización de una conexión permanente "común" a Internet durante la semana del 21 al 28 de Agosto de 2003 con el objetivo de extraer nuestras propias conclusiones al respecto.

El objetivo de la realización de este estudio es el de presentar unos datos fiables, contrastables y públicamente accesibles que nos permitan evaluar por nuestros propios métodos la situación actual así como verificar o desmentir la conveniencia de la instalación de más medidas de seguridad en las redes IP.

5.2 Red de pruebas

Una vez fijado nuestro objetivo de monitorizar una conexión a Internet de forma permanente durante una semana, debemos elegir las características que determinarán la configuración final de nuestro sistema.

Primero realizaremos una enumeración de los distintos requisitos que debe cumplir la solución propuesta. A continuación daremos forma a las posibles soluciones existentes y finalmente las evaluaremos para escoger aquella que más se ajuste a nuestras necesidades.

5.2.1 Requisitos estructurales

El objetivo de la red propuesta es el de permitir un análisis completo y exhaustivo del tráfico que circula por ella. Es por ello que existen diferentes aspectos estructurales que debemos tener en cuenta al diseñarla y que por tanto nos permitirán fijar criterios de selección entre las distintas opciones:

1. **Tipo de conexión a Internet:** Nuestro objetivo es el de monitorizar totalmente un acceso a Internet, de esta forma las características de la conexión (ubicación, acceso administrativo, ancho de banda…) determinarán la posibilidad o no de realizar esta tarea.

   Ya hemos demostrado en el capítulo de sistemas de detección de intrusos (IDS) que la monitorización en tiempo real es un tema difícil y que no siempre es posible. El ancho de banda que deseemos controlar es un factor esencial a tener en cuenta.

   Por otro lado, la monitorización de un sistema conectado a Internet requiere de privilegios de administrador en los sistemas de comunicaciones (*routers*) y los servidores, lo que limita las posibilidades a sistemas controlados por nosotros mismos.

2. **Recursos disponibles**: La disponibilidad de recursos para la realización de este trabajo no es infinita, lo que nos obliga a utilizar únicamente aquellas herramientas de las que disponemos.

   La imposibilidad de disponer de tantos equipos conectados a diferentes puntos de Internet como se desee así como la imposibilidad de pagar licencias de software debe ser tenida siempre en cuenta.

3. **Arquitectura de la red**: Para obtener unos resultados fiables en nuestro estudio debemos por un lado realizar un modelo de red genérico y parecido a la mayoría de los sistemas conectados a Internet. Por otro lado debemos realizar un diseño que nos permita el control total de la red para su monitorización sin afectar al resto de los sistemas conectados.

4. **Elementos de monitorización**: Finalmente debemos realizar un análisis de los aspectos a monitorizar así como de las herramientas existentes en el mercado que nos puedan permitir su control y análisis.

Una vez expuestos los distintos requisitos estructurales pasamos al análisis de las distintas posibilidades y opciones.

Nuestro deseo es monitorizar todo el tráfico existente en el punto de conexión a Internet seleccionado. Tal y como se observa en la figura 3-2 del capítulo de detección de intrusos, una conexión a Internet de 256Kbit puede llegar a generar hasta 2.6Gbytes de información al día, mientras que una conexión de 2Mbit generará hasta 21Gbytes diarios.

Si nuestro objetivo es la monitorización del tráfico generado en toda una semana, tenemos que la única alternativa viable es una conexión de 256Kbit (ver figura 5-1) que nos consumirá un máximo de 18Gbytes.

Tecnología **T**	Espacio de disco diario **EDD** = (T * 86400 s) / 1 GByte	Espacio en disco semanal **EDS** = EDD * 7
ADSL (256Kbits/s)	(32768 Bytes/s * 86400 s) / 1Gbyte = **2,6 Gbytes**	**18 Gbytes**
ADSL (2Mbit/s)	**21,09 Gbytes**	**147 Gbytes**

FIG. 5-1: Anchos de banda y tamaño diario de las redes más comunes.

Por otro lado, la potencia de CPU necesaria[ℜ] para el proceso de la información que circula también nos limita la posibilidad de examinar el tráfico de la red, ya que la limitación de recursos nos permite únicamente utilizar un conjunto mínimo de ordenadores con las características de PCs normales.

También deseamos que la arquitectura propuesta para el experimento sea flexible, económica y representativa de la mayoría de arquitecturas conectadas a Internet. De todas las posibles arquitecturas de conexión, la más típica y utilizada [TH96] se basa en la existencia de un router que se encarga de conectar nuestra red local (LAN) a Internet (ver figura 5-2).

[ℜ] Realmente nos referimos a potencia de procesador (CPU), disco y memoria, ya que nuestro objetivo es guardarnos todas las trazas de tráfico generadas en la red.

No entraremos en la configuración específica de la red local puesto que este aspecto no es fundamental para la realización de nuestro objetivo debido a que deseamos controlar y supervisar de todo el tráfico entre Internet y la red local. De esta forma, todas las interacciones entre elementos locales (LAN) no quedarán registradas y por tanto no afectarán a nuestro estudio.

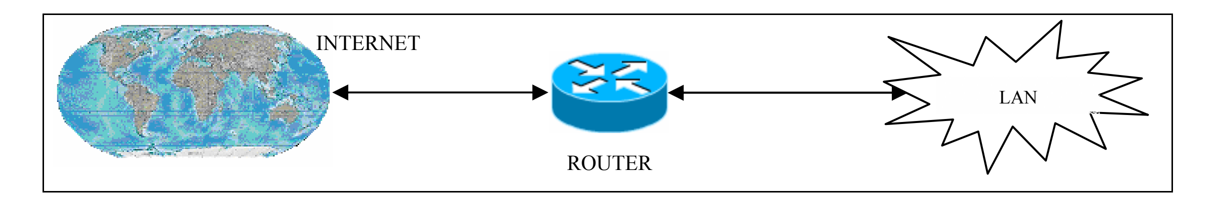


FIG. 5-2: Arquitectura de red propuesta.

5.2.2 Arquitectura propuesta

A continuación realizaremos una breve explicación de la implementación del esquema de red realizado teniendo en cuenta los distintos requerimientos funcionales anteriormente citados (ver figura 5-3).

El desglose de los elementos y características básicas de los distintos componentes utilizados en esta prueba es el siguiente:

- **Router**: Es el modelo 3COM 812 y es el encargado de conectar una línea ADSL de 256Kbit y la red local a 10Mbit.

- **HUB**: Es un modelo 3COM Dual Speed 10/100 Office Connect de 8 puertos y se encarga de conectar el router a 10Mbit con el resto de equipos de la red local a 100Mbit.

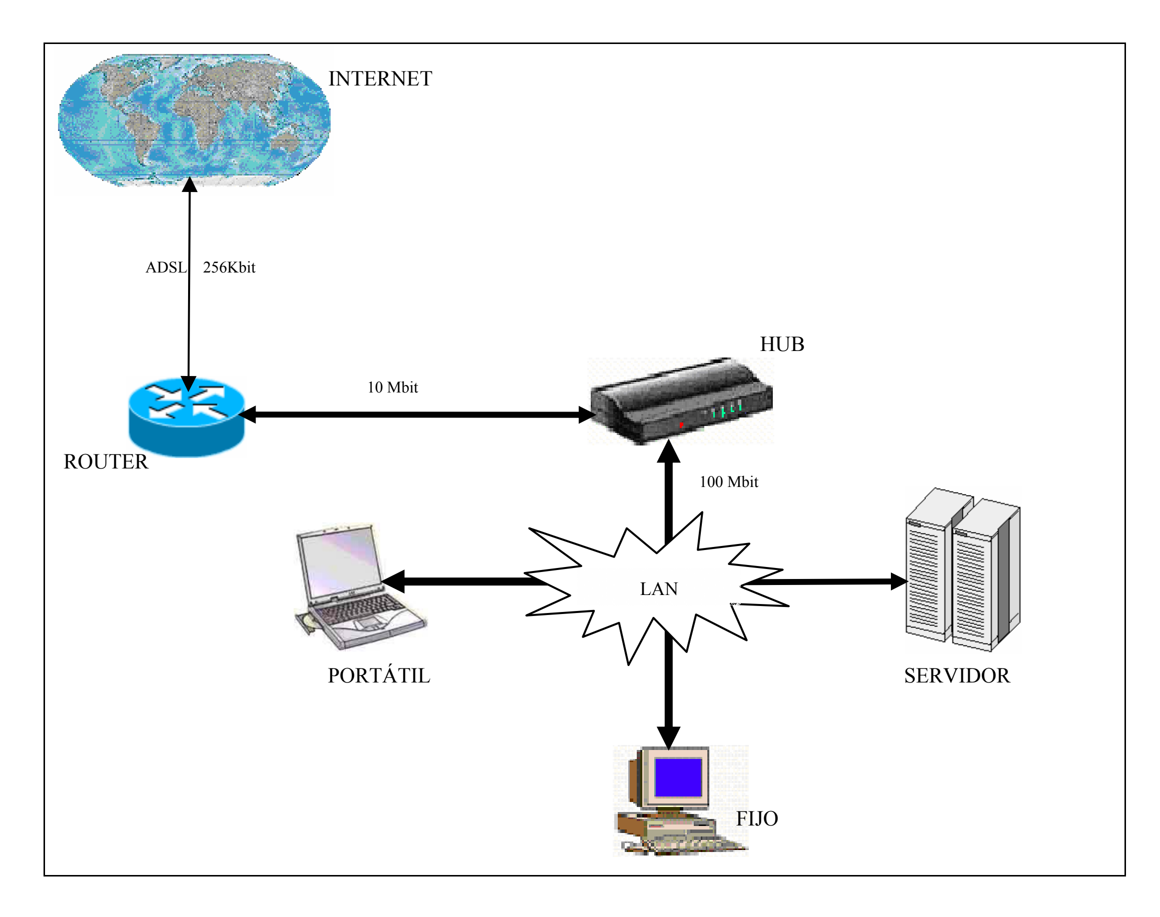


FIG. 5-3: Esquema de la red de pruebas utilizada.

- **SERVIDOR**: Es una máquina Sun Blade 100 con 512MB de RAM y dos discos duros de 120GB en RAID 1 (*mirroring*). Está configurado con el sistema operativo Linux Debian 3.0 testing y es el ordenador encargado de proporcionar los siguientes servicios:

  Servicio de resolución de nombres (*Domain Name Server, DNS*[ℜ]): Permite la resolución de nombres y direcciones IP a los usuarios de la red local. También implementa la gestión de un dominio interno para los ordenadores conectados a la LAN.

  Servicio de compartición de ficheros con Windows (*SAMBA*[℘]): Este servicio permite a los distintos usuarios acceder a los ficheros que poseen en el servidor Unix desde sistemas Windows de una forma transparente.

[ℜ] Para más información ver [Liu02][RFC1034].

[℘] Para más información ver [TCE03][WWW157].

Servicio WWW: Este servicio implementa distintas funcionalidades accesibles a los usuarios mediante un navegador. Principalmente son servicios de lectura y consulta de correo (Webmail), consulta del estado del servidor de ficheros (SAMBA/SWAT) y visualización del estado de la red.

Servicio de monitorización de red: Analiza y almacena todo el tráfico que se registra en nuestra red.

Servicio de correo: Permite a los usuarios enviar y recibir correos a Internet (SMTP) así como consultarlos mediante los protocolos IMAP4 y POP3.

SSH (*Secure SHell*): Permite la conexión segura de los usuarios al servidor.

- **FIJO**: Es un PC AMD-K6III a 450Mhz con 400MB de RAM y un disco duro de 40GB. Está configurado con un sistema operativo Windows 2000 y se utiliza como puesto de trabajo.

- **PORTÁTIL**: Es un PC portátil con un INTEL PIV a 2.4Ghz con 512MB y 40GB de disco duro. Está configurado con un sistema operativo Windows XP y se utiliza como puesto de trabajo móvil.

5.2.3 Configuración

Una vez presentada la arquitectura de red que utilizaremos, pasaremos a pormenorizar los distintos aspectos técnicos de su configuración.

Debido a que el tipo de conexión a Internet utilizado (ADSL) únicamente proporciona una dirección IP pública, se ha tenido que realizar una configuración específica en el router que permita:

1. **Acceso a Internet a todos los ordenadores de la red local (LAN) mediante el uso de técnicas de NAT**[ℜ]: Los ordenadores conectados a la LAN reciben automáticamente direcciones IP del rango 192.168.0.XXX mediante el protocolo DHCP[℘] que implementa el router. De esta forma, todas las direcciones locales tienen acceso a Internet.

2. **Redirección del tráfico de red generado desde y hacia la red local al ordenador encargado de la monitorización del sistema** (servidor): Debido a que deseamos examinar todo el tráfico que llega a nuestra dirección pública, el router ha sido configurado de forma que automáticamente redirija (*bridging*) todo el tráfico de Internet no solicitado por los ordenadores locales al ordenador encargado del análisis.

En la configuración propuesta se utiliza un HUB para la interconexión de los elementos internos de la LAN (router y ordenadores). Esta decisión no es arbitraria, sino que responde a la característica que tienen todos los HUBs de reenviar lo que reciben por un puerto a todos los demás. De esta forma, nos aseguramos que todo el tráfico existente en la red local (vaya o no específicamente al ordenador "servidor") llegará a él.

Las distintas herramientas de monitorización y control del tráfico de red utilizan un modo de trabajo de las tarjetas de red denominado "**modo promiscuo**". El modo de funcionamiento normal de una tarjeta de red consiste en que al recibir un paquete de información, si la dirección de destino no es el la que hay configurada en la tarjeta de red lo ignora. En el modo promiscuo toda la información que llega a la tarjeta de red es accesible.

De esta forma, con la configuración realizada (usando un HUB en lugar de un SWITCH) podemos realmente controlar todo el tráfico existente en nuestra red. Además, durante la semana en la que se realizó este experimento únicamente se mantuvo en funcionamiento el ordenador encargado de la recolección de datos (servidor), de esta forma evitamos cualquier tipo de interferencia que pudieran causar las conexiones de los otros ordenadores locales.

[ℜ] Network Adderss Translation, para más información consultar la bibliografía [Has97].
[℘] Dinamic Host Configuration Protocol, para más información mirar bibliografía [DLD02][WWW156].

Por otro lado, el hecho de mantener un sistema conectado a Internet únicamente "escuchando" y sin generar ningún tipo de tráfico concreto (solamente las respuestas a peticiones iniciadas desde Internet) permite que los resultados obtenidos puedan extrapolarse a cualquier otro sistema conectado a Internet ya que:

- Tenemos un punto de conexión a Internet que no genera tráfico intrínsecamente, lo que nos permite asegurar que si recibimos tráfico externo, cualquier otro nodo conectado a Internet **puede** recibirlo también.

- La elección del periodo de tiempo de análisis (una semana) viene determinada principalmente por tres factores:

    1. **Repetición de los patrones obtenidos:** La observación del tráfico en un día es insuficiente para poder extraer conclusiones plausibles, mientras que la observación en varios días puede permitir la correlación de resultados.

    2. **El tamaño de los datos en disco:** Cada semana se podrían llegar a generar hasta 18Gbytes de información (tope teórico de transmisión), cantidad mas que suficiente para analizar y realizar pruebas.

    3. **Seguridad:** La realización de este experimento requiere de una supervisión constante del sistema ya que cualquier ataque puede resultar exitoso y comprometer nuestro sistema. El esfuerzo de supervisión del sistema por largos períodos de tiempo queda fuera de nuestros recursos.

5.3 Herramientas

En este apartado realizaremos una breve explicación de las características principales y rol asumido dentro de este experimento práctico de las diferentes herramientas de software elegidas.

Una vez decidido nuestro objetivo se procedió a la selección de las herramientas necesarias para llevarlo a cabo. Los criterios en los que nos hemos basado para la selección de estos programas son los siguientes:

- **Plataforma de funcionamiento**: Las herramientas elegidas deben funcionar perfectamente en la mayoría de plataformas existentes.

- **Licencia de uso**: La licencia bajo la que se distribuya el software debe permitir su uso de forma libre y sin limitaciones contractuales en nuestro ámbito de estudio (experimento). Herramientas distribuidas bajo licencias GPL, GNU, BSD o similares serán las candidatas.

- **Disponibilidad del código fuente**: Los programas seleccionados deberán estar disponibles en forma de código fuente. Parte de este experimento es la compilación de las herramientas que se van a utilizar. **NO** se utilizarán binarios precompilados en otros sistemas.

- **Continuidad del proyecto**: Siempre que sea posible la elección entre varias herramientas similares, la continuidad del proyecto (existencia de más de un desarrollador, que el software tenga mas de nueve meses de vida…) será una prioridad. Esto nos asegura que la herramienta seleccionada nos permitirá tener soporte de sus creadores en caso de necesidad.

- **Madurez del software**: Se buscan herramientas estables que ya tengan desarrolladas varias versiones. Nuestro objetivo es el de evitar el uso de herramientas en fase de desarrollo o poco probadas que puedan introducir inestabilidad en el sistema.

- **Integración con el sistema Unix**: El software elegido debe ser fácilmente integrable en plataformas Unix (Solaris, Linux, AIX…) y preferiblemente en otros sistemas existentes (Windows…). Análogamente, los prerrequisitos que puedan necesitar las diferentes herramientas deberán de ser lo más standard posible.

5.3.1 APACHE

Para las funciones de servidor WWW (principalmente visualización de páginas HTML de los programas instalados) utilizaremos el software **Apache** [WWW166] versión 1.3.27.

Apache el servidor WWW líder con más de un 63% [WWW167] de cuota. Es un proyecto maduro que lleva varios años produciendo un software de calidad y gratuito que funciona perfectamente en casi cualquier sistema operativo existente (desde Windows hasta sistemas Unix).

La compilación e instalación de este software es la usual (por defecto) que recomienda el sistema. Las únicas modificaciones que se han tenido que realizar para adecuarlo al resto de programas que hemos utilizado, son la configuración del servidor WWW para que permita servir páginas y ejecución de *scripts* a los usuarios locales del sistema[ℜ] (ver figura 5-4).

Añadir al fichero “**httpd.conf**”:

```
#
# UserDir: The name of the directory which is appended onto a user's home
# directory if a ~user request is received.
#

<IfModule mod_userdir.c>
   UserDir public_html
</IfModule>

<Directory /home/*/public_html/cgi-bin>
     Options +ExecCGI -Includes -Indexes
     SetHandler cgi-script
 </Directory>
```

FIG. 5-4: Modificación del archivo de configuración de Apache.

[ℜ] Las expresiones del tipo http://servidor/~usuario acceden a los archivos del usuario situados en /home/usuario/public_html/ análogamente la ejecución de *scripts* http://servidor/~usuario/cgi-bin accede a /home/usuario/public_html/cgi-bin/

5.3.2 Ethereal

Para la visualización y manipulación del tráfico de la red capturado utilizaremos el programa **Ethereal** [WWW168] versión 0.9.14.

Este programa goza de gran reputación entre los administradores de red y se ha convertido en una herramienta básica para cualquier análisis de redes. Este software es gratuito y funciona perfectamente tanto en sistemas Unix como Windows.

Entre sus características principales se encuentra la de realizar completos análisis y visualizaciones en tiempo real de todos los protocolos de redes más usuales. También permite la captura del tráfico de red en disco y la aplicación de filtros de selección así como la visualización del contenido de los datagramas, su edición, modificación y selección interactiva en tiempo real (ver figura 5-5).

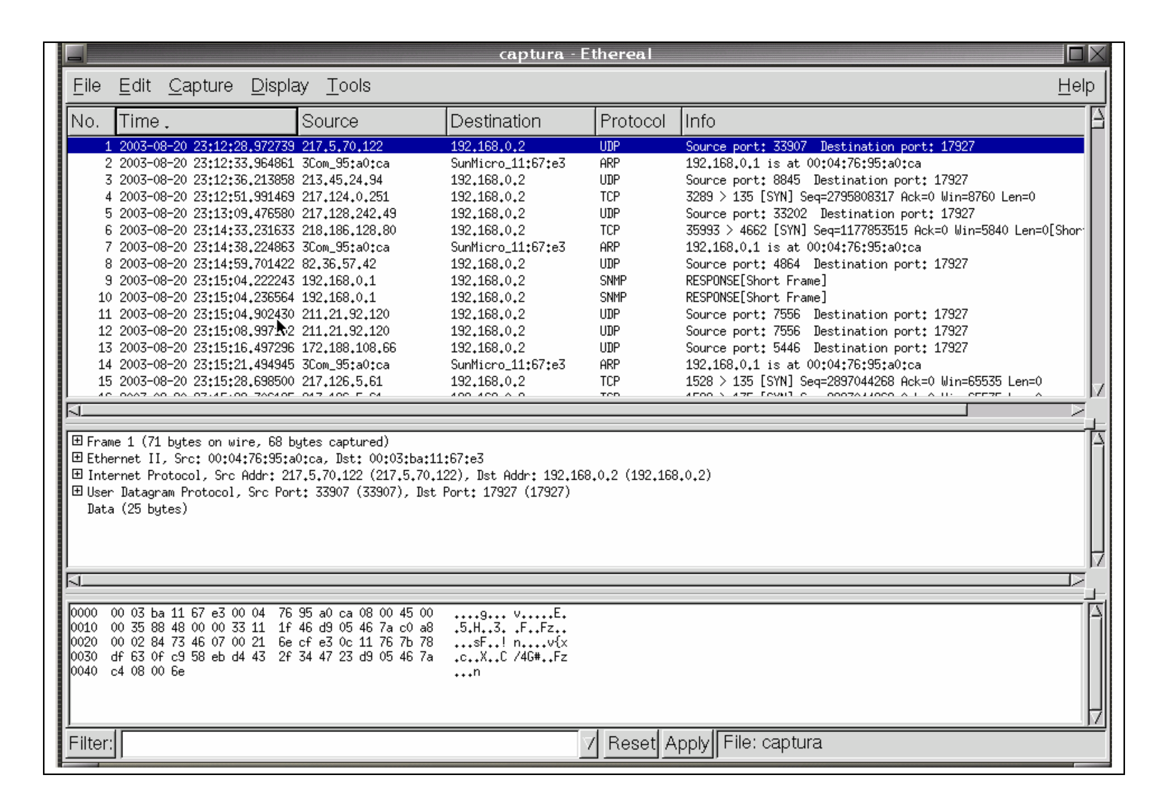


FIG. 5-5: Ethereal.

La compilación e instalación de este software es la que se realiza por defecto. No obstante se han de tener en cuenta los siguientes requisitos:

1. Necesita las librerías GTK+ para su entorno gráfico.

2. Necesita las librerías del sistema GLIB.

3. Necesita las librerías de interface LIBPCAP para su acceso a los dispositivos de red.

4. Para poder utilizar todas sus funcionalidades correctamente debe ser ejecutada como administrador (*root*) del sistema.

Estos requisitos no son excesivos o complicados, ya que usualmente cualquier sistema lleva instaladas estas librerías por defecto. En cuanto a la necesidad de ser administrador de la máquina es debido a que el programa trata directamente con el dispositivo de red configurándolo en modo promiscuo para poder espiar todos los paquetes de información que llegan al dispositivo.

5.3.3 IPaudit- IPaudit WEB

Para la visualización y la generación de las gráficas del tráfico recibido en nuestra red utilizaremos el programa **IPaudit** [WWW169]. Este programa dispone de una versión con interface WWW denominada **IPaudit WEB** [WWW170] que será la que se utilizará en su versión 1.0-BETA7.

La elección de IPaudit sobre otras herramientas de monitorización de redes se basa en que cumple perfectamente los requisitos estructurales planteados inicialmente y existe una gran experiencia en su uso por nuestra parte.

Es un proyecto sólido (empezó en 1999) que ha ido evolucionando gradualmente y dispone de un grupo de gente que le proporciona continuidad. Funciona perfectamente en entornos Unix y proporciona una gran flexibilidad de configuración. Además genera una gran variedad de informes útiles que nos permiten conocer el estado real de nuestra red.

También tiene la capacidad de realizar complejas búsquedas (por dirección IP de origen, destino, protocolo…) por los distintos logs que genera. Asimismo almacena y realiza las búsquedas de los datos en formato comprimido (con el programa *gzip* concretamente) lo que permite un gran ahorro de espacio de disco, vital en este tipo de aplicaciones.

Hemos escogido el uso de la versión IPaudit WEB debido principalmente a que la versión “normal” de IPaudit es en modo texto (línea de comandos) lo que dificulta la interpretación de los datos. Por otro lado, la versión WWW nos permite consultar de una forma visual sencilla los diferentes datos y gráficas generadas por el tráfico de red (ver figura 5-6).

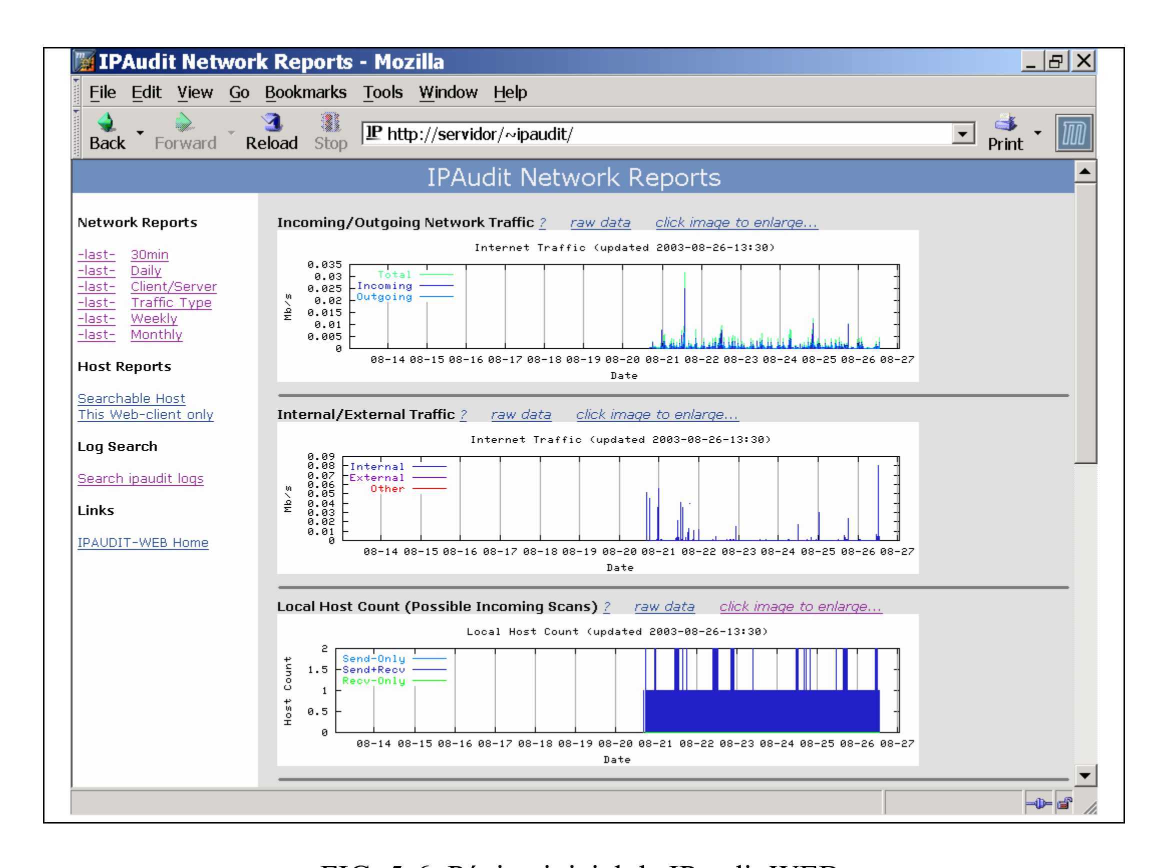


FIG. 5-6: Página inicial de IPaudit WEB.

La versión WEB utiliza procesos CRON que se lanzan cada media hora y que durante treinta minutos se encargan de registrar todo el tráfico generado en la red. De esta forma tenemos que cada 30 minutos se envía una señal de finalización controlada al proceso anterior (que escribe en el fichero de log correspondiente los datos del tráfico pendientes) y se inicia un nuevo proceso de captura de datos.

Esta captura de datos se almacena en disco aplicando un filtro de conversión a un formato más compacto que incluye los campos más relevantes de los paquetes recibidos (ver figura 5-7). De esta forma podemos observar como si bien tenemos todos los datos que determinan la comunicación entre las distintas direcciones IP, no disponemos de los datos transmitidos[ℜ] (el contenido).

```
LOCAL-IP
|               REMOTE-IP
|               |               PROTOCOL
|               |               | LOCAL-PORT
|               |               | |    REMOTE-PORT
|               |               | |    |    INC-BYTES
|               |               | |    |    |      OUT-BYTES
|               |               | |    |    |      |      INC-PACK
|               |               | |    |    |      |      |   OUT-PACK
|               |               | |    |    |      |      |   |   FIRST-TIME
|               |               | |    |    |      |      |   |   |             LAST-TIME
|               |               | |    |    |      |      |   |   |             |             FIRST-HOST
|               |               | |    |    |      |      |   |   |             |             | LAST-HOST
|               |               | |    |    |      |      |   |   |             |             | |
192.168.128.123 068.062.079.141 6 1581 1214 84573 51867 601 609 18:30:00.8251 18:35:41.3365 1 2
192.168.128.123 068.042.071.248 6 1445 1214 276031 333379 2178 2277 18:30:00.8251 19:00:00.1441 1 1
192.168.128.123 068.113.229.170 6 1275 1214 254045 279761 2011 2030 18:30:00.8251 19:00:00.4148 1 1
192.168.128.123 068.114.252.038 6 1654 1214 302931 304077 2725 2829 18:30:00.8251 19:00:00.4200 1 1
192.168.128.123 068.049.032.236 6 1452 1214 330275 312609 3010 3180 18:30:00.8251 19:00:00.4332 1 1
```

FIG. 5-7: Página inicial de IPaudit WEB.

Una vez finalizado el proceso anterior se generan las gráficas y los informes correspondientes:

- **Informes cada 30 minutos**: En este tipo de informe se muestra el total de bytes enviados y recibidos en nuestra red durante la media hora analizada. Asimismo se muestran las veinte direcciones IP locales y remotas con más tráfico generado, lo que nos permite examinar más a fondo las comunicaciones que nos interesen (ver figura 5-8).

[ℜ] Para obtener los datos que se intercambian durantes las comunicaciones de red debemos usar otros programas como el Ethereal o el tcpdump.

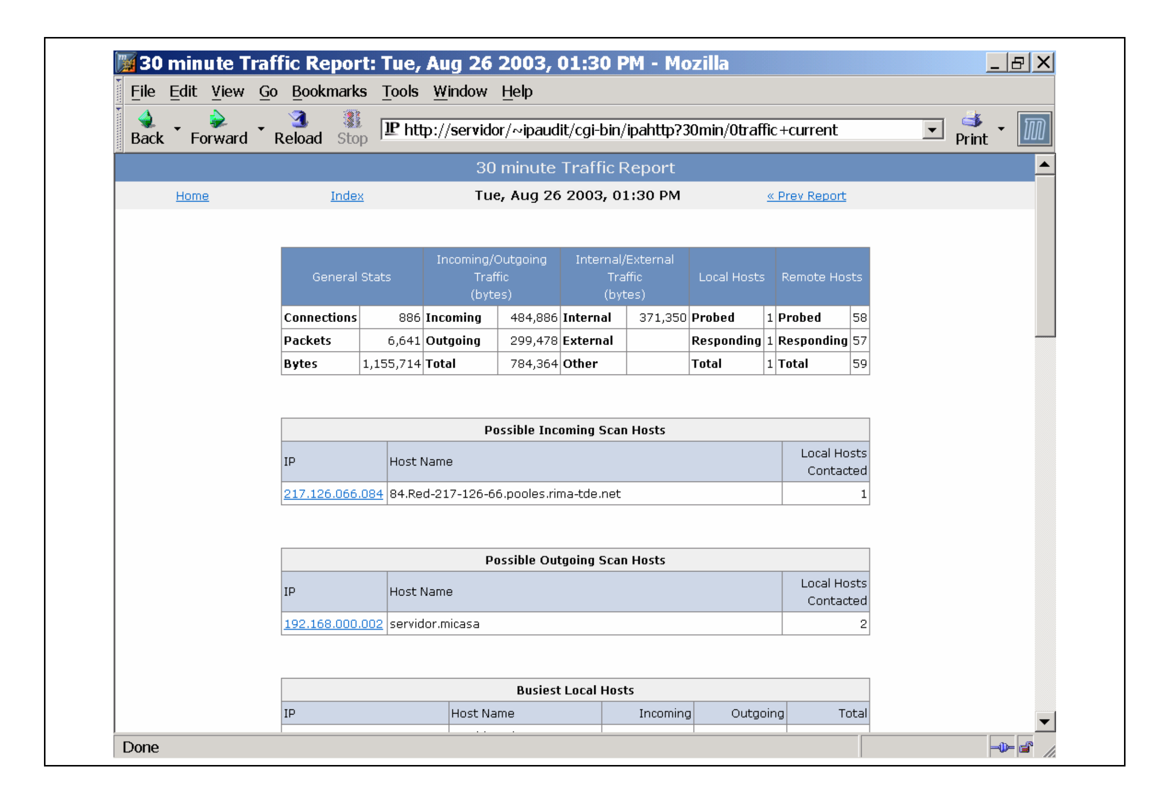


FIG. 5-8: Informe de IPaudit WEB de los 30minutos.

- **Informes diarios**: Estos informes presentan el resumen de la acumulación de todos los informes de media hora realizados durante el día. Visualiza el total de bytes enviados y recibidos en nuestra red así como las veinte direcciones IP locales y remotas que más tráfico han generado o recibido (ver figura 5-9).

- **Informes cliente/servidor** (diario): Este informe se genera diariamente y desglosa todo el tráfico recibido o enviado a los servidores que proporcionan los servicios de Mail, SSH, Telnet, HTTP, HTTPS tanto para servidores o clientes locales como remotos (ver figura 5-10).

- **Informes por tipo de tráfico** (diario): En este informe se realiza una agregación de todo el tráfico generado diariamente y se clasifica según el protocolo (TCP, UDP, ICMP, NetBios, Telnet, FTP…) al que pertenezca (ver figura 5-11).

- **Informes semanales**: Estos informes reflejan la suma total del tráfico generado por la red durante la semana y presentan las 25 direcciones IP que más tráfico han generado o recibido (ver figura 5-12).

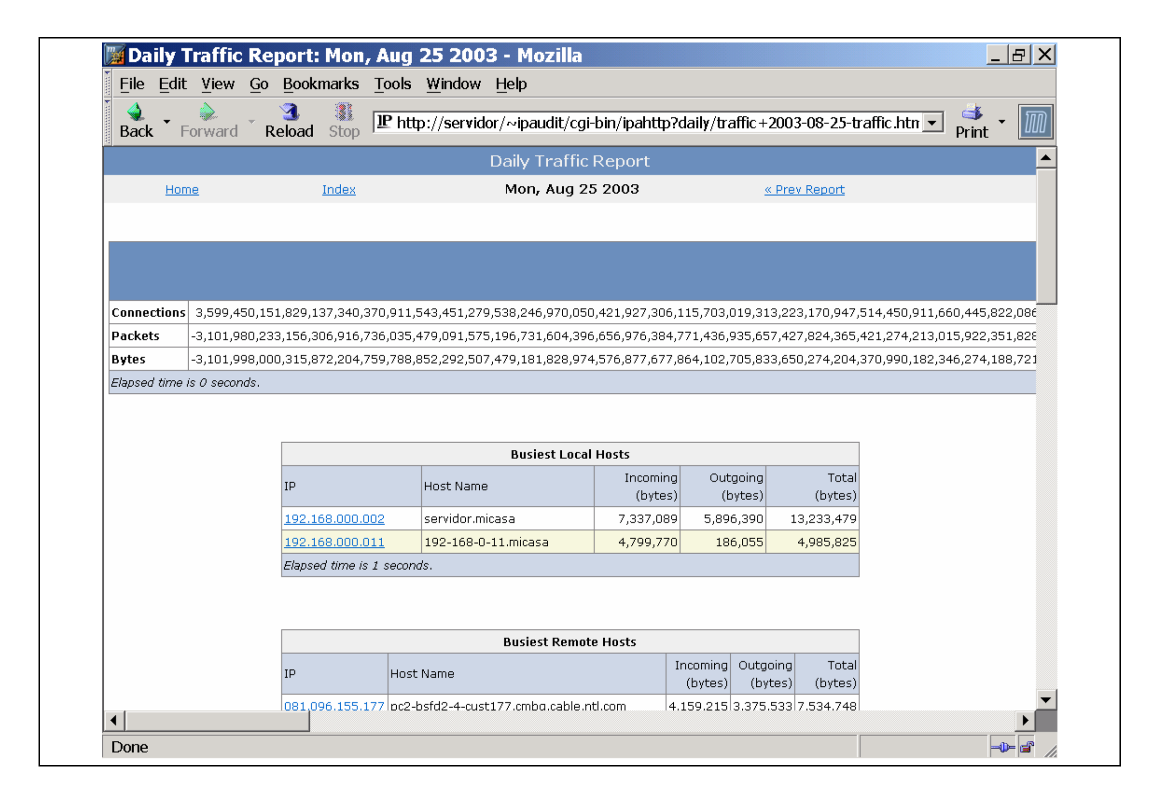


FIG. 5-9: Informe de IPaudit WEB diario.

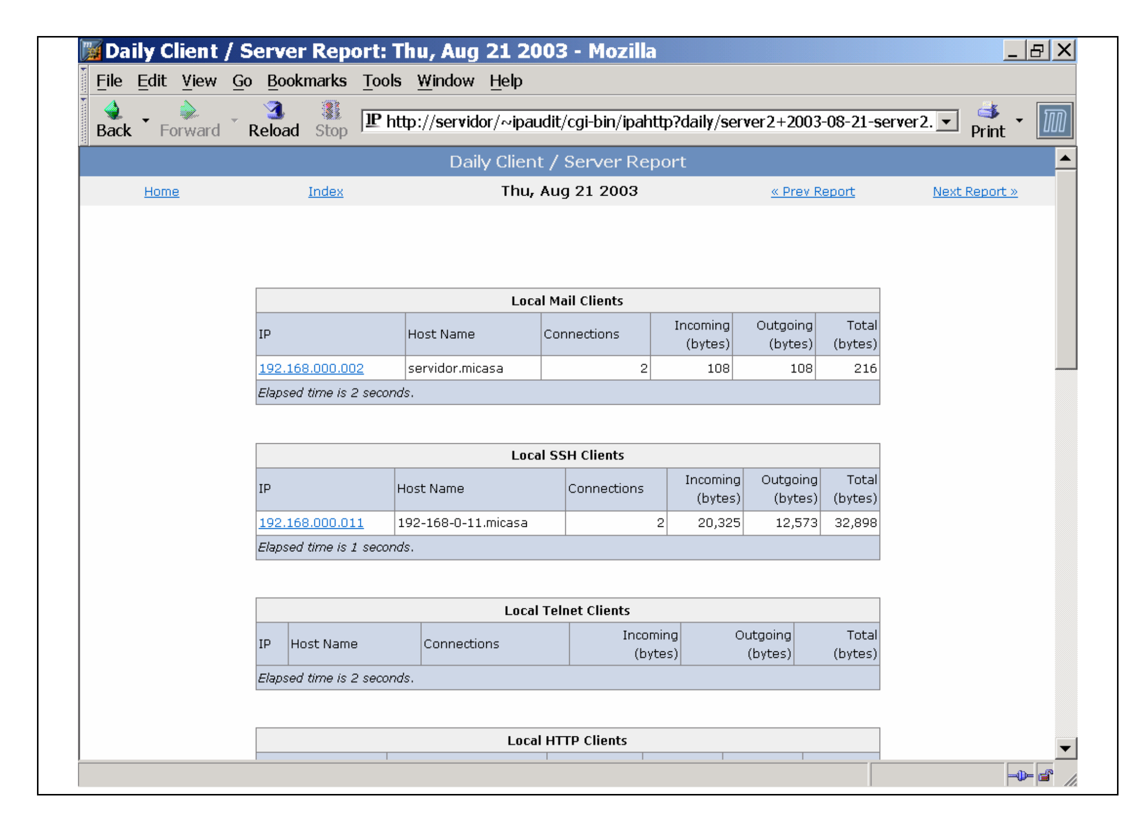
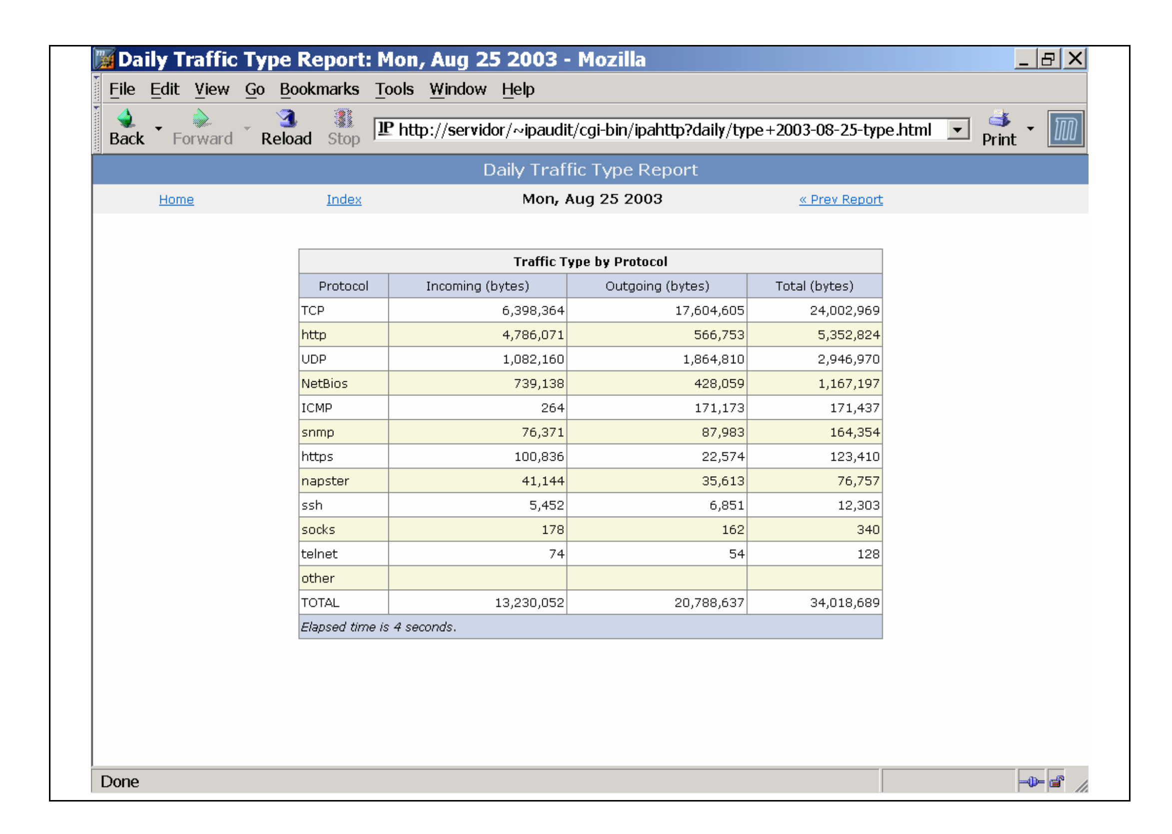


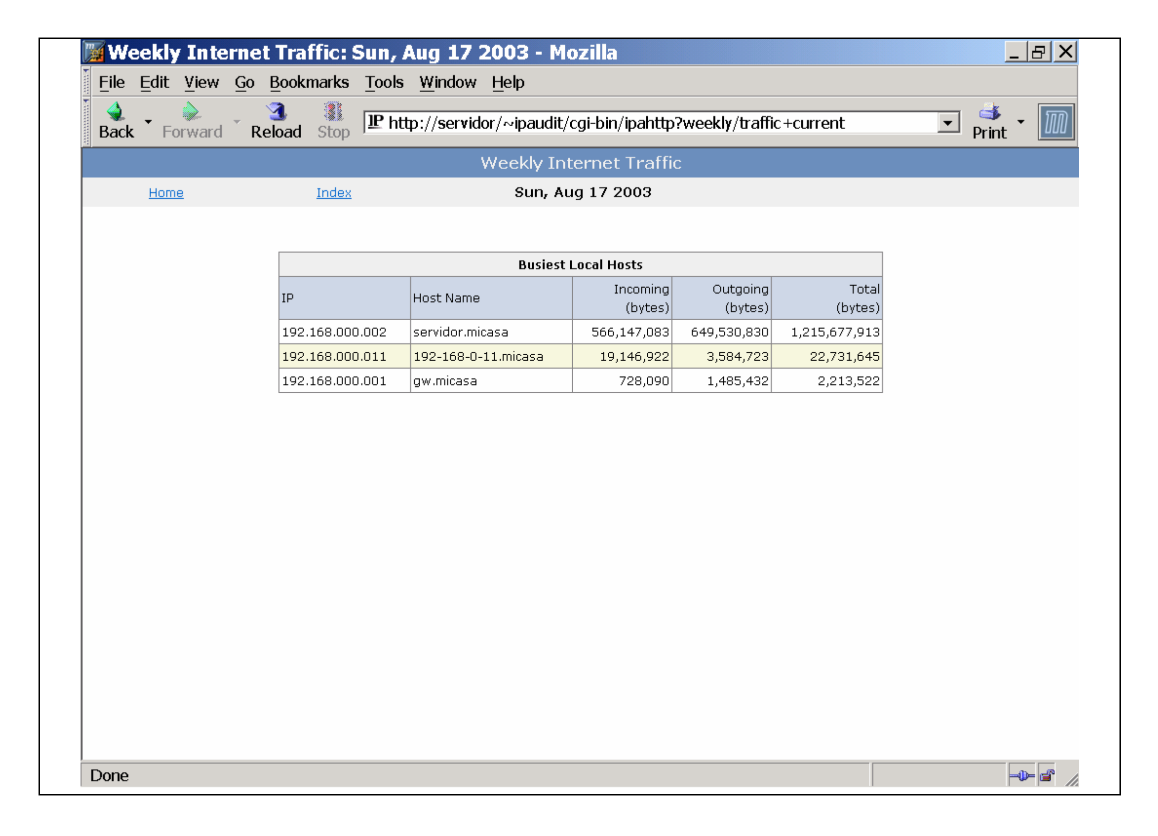
FIG. 5-10: Informe de IPaudit WEB diario cliente/servidor.



Traffic Type by Protocol			
Protocol	Incoming (bytes)	Outgoing (bytes)	Total (bytes)
TCP	6,398,364	17,604,605	24,002,969
http	4,786,071	566,753	5,352,824
UDP	1,082,160	1,864,810	2,946,970
NetBios	739,138	428,059	1,167,197
ICMP	264	171,173	171,437
snmp	76,371	87,983	164,354
https	100,836	22,574	123,410
napster	41,144	35,613	76,757
ssh	5,452	6,851	12,303
socks	178	162	340
telnet	74	54	128
other			
TOTAL	13,230,052	20,788,637	34,018,689

Elapsed time is 4 seconds.

FIG. 5-11: Informe de IPaudit WEB diario por tipo de tráfico.



Busiest Local Hosts				
IP	Host Name	Incoming (bytes)	Outgoing (bytes)	Total (bytes)
192.168.000.002	servidor.micasa	566,147,083	649,530,830	1,215,677,913
192.168.000.011	192-168-0-11.micasa	19,146,922	3,584,723	22,731,645
192.168.000.001	gw.micasa	728,090	1,485,432	2,213,522

FIG. 5-12: Informe de IPaudit WEB semanal.

- **Informes mensuales**: Este informe es exactamente igual que el informe semanal, pero mostrando el tráfico de las 25 direcciones IP con más tráfico de red durante el mes.

IPaudit WEB también permite la realización de búsquedas en los ficheros de log por una gran cantidad de campos (tipo de protocolo, dirección IP, puerto de origen, puerto de destino…) lo que permite filtrar el tráfico registrado para analizar únicamente las comunicaciones deseadas (ver figura 5-13).

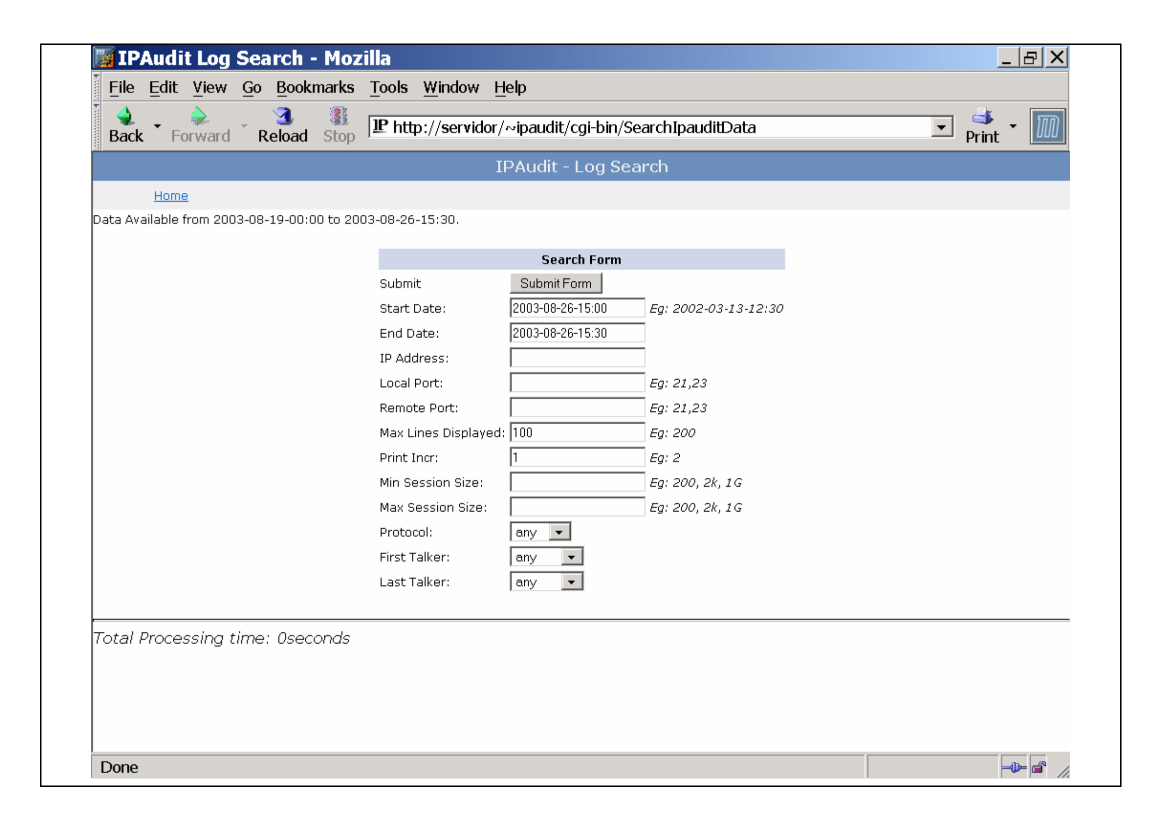


FIG. 5-13: Búsquedas en IPaudit WEB.

La instalación efectuada es la que se recomienda por defecto. Sin embargo se han de tener en cuenta los siguientes requisitos:

1. Necesita las librerías de interface LIBPCAP para su acceso a los dispositivos de red.

2. Necesita la utilidad GNUPLOT para la generación de los gráficos.

3. Es necesario el compilador del lenguaje PERL debido a que IPaudit WEB ejecuta varios *scripts* escritos en este lenguaje.

4. Necesita que se cree un usuario específico en el sistema denominado "*ipaudit*" y privilegios de administrador para realizar la instalación del software.

5. Es necesario un servidor WWW que esté configurado para que permita al usuario ipaudit la visualización de páginas HTML (http://servidor/~ipaudit) y la ejecución de *scripts* (http://servidor/~ipaudit/cgi-bin/) desde su directorio (/home/ipaudit/public_html usualmente).

6. El sistema debe permitir la ejecución planificada (CRON) de *scripts* a los usuarios. Este punto es esencial ya que el programa genera gráficas cada cierto intervalo de tiempo, lo que le obliga a ejecutar determinados comandos cada 30 minutos.

5.3.4 MRTG

MRTG (*Multi Router Traffic Grapher*) [WWW171] es una herramienta gráfica que permite la monitorización de las conexiones de red mediante la generación de gráficas que reflejan el uso del ancho de banda. Se comunica con los distintos dispositivos mediante el protocolo SNMP[ℜ] y suele utilizarse en la monitorización de equipos de red, principalmente en routers y swiches. La versión que se ha utilizado en este experimento ha sido la 2.9.29.

MRTG es una herramienta gratuita, estable y que funciona prácticamente con cualquier sistema Unix. En nuestro esquema de red este software será el encargado de la monitorización del ancho de banda que circula por el router, tanto del extremo conectado a Internet (ADSL) como del extremo conectado a la red local (LAN).

ℜ Simple Network Management Protocol [Ric98-1].

A diferencia de otras herramientas de monitorización como el IPaudit, MRTG simplemente genera una gráfica diaria, semanal, mensual y anual con el total del ancho de banda utilizado en la conexión a Internet (ver figura 5-14). De esta forma, es el complemento perfecto para el IPaudit, ya que nos permite obtener un control exhaustivo de los dos segmentos de la red que conecta el router.

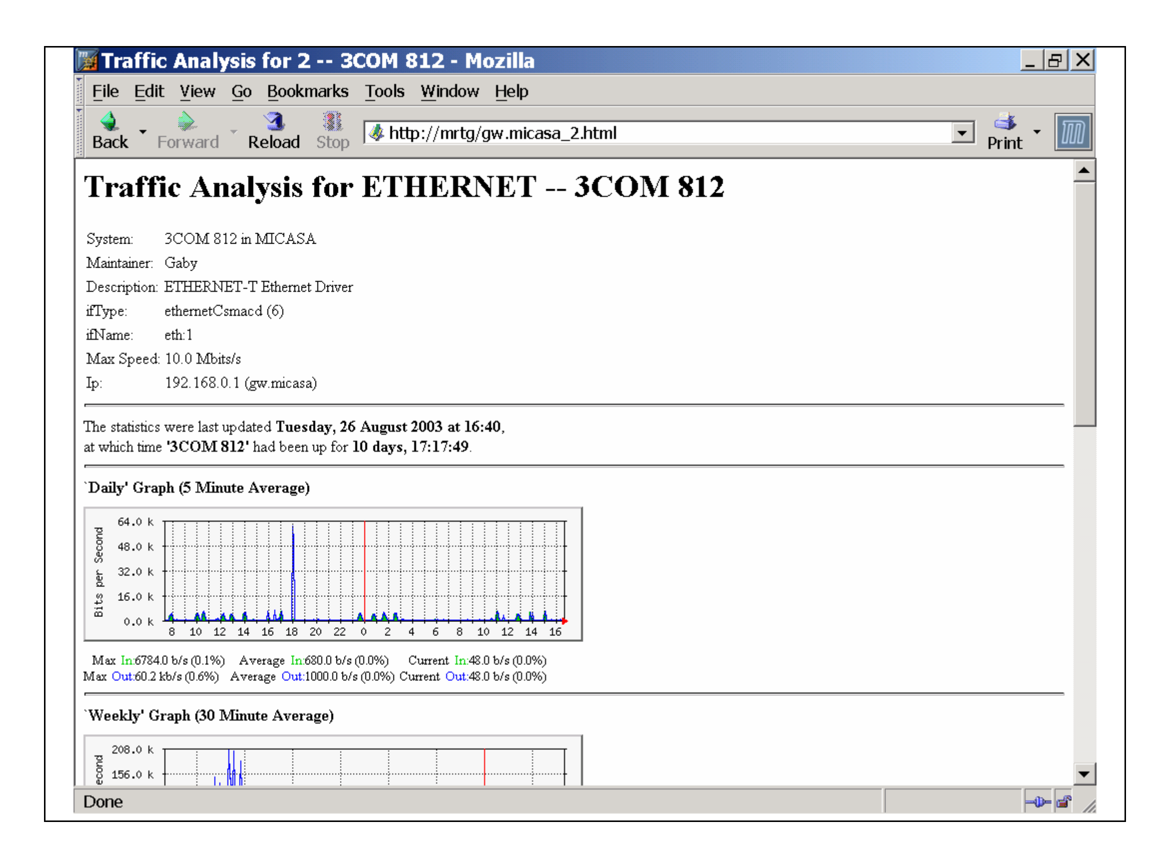


FIG. 5-14: MRTG.

La instalación efectuada es la que se recomienda por defecto. Sin embargo se han de tener en cuenta los siguientes requisitos:

1. Los dispositivos monitorizados con MRTG deben tener activado el protocolo SNMP para que el software pueda conectarse a ellos y leer los datos.

2. Es necesario un servidor WWW.

3. Son necesarias las librerías LIBPNG que permiten la creación y manipulación de imágenes en formato PNG.

4. El sistema debe permitir la ejecución planificada (CRON) de *scripts* a los usuarios. Este punto es esencial ya que el programa genera gráficas cada cierto intervalo de tiempo, lo que le obliga a ejecutar determinados comandos cada 5 minutos.

5.3.5 NMAP

NMAP (*Network Mapper*) [WWW172] es una herramienta que permite la exploración y la auditoria de redes de ordenadores. Su misión principal consiste en la realización de varios tipos de exámenes de puertos (*port scanning*) para la obtención de los servicios existentes en un ordenador. La versión que se ha utilizado en este experimento ha sido la 3.3.0.

Este software gratuito, muy estable que funciona tanto en sistemas Unix como Windows. Es ampliamente conocido y utilizado por administradores de red y *hackers*.

El objetivo del uso de esta herramienta en nuestra arquitectura es la de verificar los servicios accesibles desde nuestra red hacia Internet. De esta forma, podemos comprobar la eficacia de nuestro sistema de monitorización y asegurarnos que cualquier tipo de tráfico que se genere desde Internet quedará registrado.

Otra característica de NMAP es la de efectuar la detección del sistema operativo existente en el ordenador (*OS fingerprinting*). De esta forma, antes de abrir nuestra red a Internet podemos auditarnos de forma que ya sabremos toda la información que cualquiera podría llegar a extraer de nosotros.

La instalación efectuada es la que se recomienda por defecto. Sin embargo cabe notar que para ejecutar todas las funcionalidades que proporciona esta herramienta se debe tener privilegios de administrador (*root*).

5.3.6 TCPdump

TCPdump [WWW173][WWW174] es una herramienta que permite la auditoria y la adquisición del tráfico que circula por la red. La versión que se ha utilizado en este experimento ha sido la 3.7.2.

Este software es gratuito y funciona sin problemas tanto en sistemas Unix como Windows. Pertenece a un proyecto estable que lleva varios años desarrollando software de calidad y se le considera como el referente principal en el análisis del tráfico de redes.

TCPdump permite el uso de filtros de tráfico (por protocolo, por dirección IP de origen o destino, por puerto de origen o destino…) así como diferentes opciones de captura tanto de las cabeceras de los paquetes como de los datos que transportan. Precisamente esta característica es la que utilizaremos principalmente.

Otros programas de monitorización de redes capturan en mayor o menor medida el tráfico existente en la red pero únicamente a nivel de cabeceras (como IPaudit) o a nivel de tamaño del paquete (como MRTG). TCPdump nos permite almacenar junto con la cabecera del datagrama los datos que transporta, lo que nos permite la reconstrucción total de las comunicaciones existentes en nuestra red (ver figura 5-15).

- Ejemplo de la captura del tráfico del dispositivo ETH0 de nuestra red en el fichero 'captura.cap':

```
servidor:/home/gaby/tcpdump/sbin# ./tcpdump –i eth0 –ln -w captura.cap
tcpdump: listening on eth0
```

FIG. 5-15: Ejemplo de captura de datos con TCPdump.

Los ficheros de log de salida generados por TCPdump son ficheros binarios que pueden ser visualizados/manipulados por otras herramientas como el Ethereal.

La instalación efectuada es la que se recomienda por defecto. Sin embargo se han de tener en cuenta los siguientes requisitos:

1. Necesita las librerías de interface LIBPCAP para su acceso a los dispositivos de red.

2. Para ejecutar todas las funcionalidades que proporciona esta herramienta se deben tener privilegios de administrador (*root*). Esto es debido principalmente a que accede directamente al dispositivo de red y lo configura en modo promiscuo para la captura de todo el tráfico.

5.3.7 TCPreplay

TCPreplay [WWW175] es una herramienta que permite "a posteriori" la reproducción del tráfico de red a partir de los logs generados por el programa TCPdump o compatibles. La versión utilizada en las pruebas realizadas es la 1.4.4.

El proyecto TCPreplay es gratuito y está diseñado para funcionar perfectamente en sistemas Unix.

Después de monitorizar y capturar todo el tráfico de nuestra red, puede interesarnos realizar una simulación de algún comportamiento anómalo detectado o que deseemos analizar en profundidad.

Con TCPdump podemos aplicar los filtros deseados a la captura del tráfico de red, mientras que con Ethereal podemos visualizarlo y modificarlo. TCPreplay cierra el círculo permitiéndonos la reproducción de secuencias de tráfico ya capturadas en un entorno controlado. Además nos permite la posibilidad de reproducir estas secuencias tantas veces como queramos y a la velocidad que deseemos, lo que permite simular distintos comportamientos según el ancho de banda configurado.

La instalación efectuada es la que se recomienda por defecto. Sin embargo se han de tener en cuenta los siguientes requisitos:

1. Necesita las librerías de interface LIBPCAP para su acceso a los dispositivos de red.

2. Necesita las librerías de interface LIBNET para su acceso a los dispositivos de red.

3. Para ejecutar todas las funcionalidades que proporciona esta herramienta se deben tener privilegios de administrador (*root*).

5.4 Resultados

Los resultados presentados a continuación hacen referencia al tráfico observado durante la semana del 21 al 28 de Agosto de 2003. La presentación de los datos se desglosará en dos bloques.

Inicialmente se realizará una presentación de los datos mediante un informe diario que contendrá los datos más relevantes del día así como su análisis. Después se realizará un resumen semanal que contendrá los datos más importantes del tráfico observado durante la semana analizada.

Dentro del informe diario se realizarán dos presentaciones distintas de los datos obtenidos en nuestra red.

La primera clasificación dividirá en tráfico dirigido según los puertos a los que haga referencia. De esta forma tendremos los inferiores al 1024 denominados conocidos[ℜ] (*well-know ports*) desglosados por servicios y la dirigida a puertos iguales o superiores al 1024:

[ℜ] Según la clasificación de la IANA [WWW176][WWW177].

1. **MAIL**: En esta categoría se clasificaría todo el tráfico que hiciera referencia a los servicios de correo más usuales (SMTP, POP3 e IMAP). Puertos de conexión 25, 110 y 143.

2. **SSH**: En este grupo se colocarán las peticiones recibidas al servicio de conexión remota segura. Puerto de conexión 22.

3. **TELNET**: Este grupo hará referencia a las peticiones recibidas al servicio de conexión remota. Puerto de conexión 23.

4. **HTTP**: En esta categoría se incluye todo el tráfico destinado al servidor WWW. Puerto de conexión 80.

5. **HTTPS**: En esta categoría se incluye todo el tráfico destinado al servidor WWW seguro. Puerto de conexión 443.

6. **Netbios-ns**: En este grupo se indicarán las peticiones recibidas al servicio de resolución de nombres (*name server*) de *Netbios*. Puerto de conexión 137.

7. **Netbios-dmg**: Este grupo abarca las peticiones recibidas al servicio de datagramas (*datagram service*) de *Netbios*. Puerto de conexión 138.

8. **Netbios-ssn**: Esta categoría referenciará todas las peticiones recibidas al servicio de sesión (*session service*) de *Netbios*. Puerto de conexión 139.

9. **Microsoft-ds**: En este grupo se indicarán las peticiones recibidas al servicio de Microsoft DS. Puerto de conexión 435

10. **Otros (< 1024)**: En esta categoría se incluirá todo el tráfico destinado a otros puertos inferiores al 1024 y que no corresponde a ninguno de los grupos anteriores.

11. **Otros (>= 1024):** En esta última categoría se clasificará todo el tráfico destinado a puertos superiores al 1024 (puertos no standard).

La segunda clasificación agrupará los datagramas recibidos según el tipo de protocolo utilizado para su transmisión. Análogamente los protocolos en los que subdividiremos el tráfico obtenido son los más usuales en Internet:

- **TCP**: Protocolo orientado a conexión y fiable. Lo utilizan servicios como el SSH, TELNET, WWW…

- **UDP**: Protocolo no fiable. Lo utilizan servicios como el DNS, NFS…

- **ICMP**: Protocolo no fiable utilizado para la gestión y el control del flujo de las comunicaciones IP. Lo utilizan servicios como en PING, Traceroute…

- **Otros**: Agrupa el resto de tráfico que no haga referencia a ninguno de los protocolos anteriores.

Para el análisis "fino" de los datos obtenidos con el programa IPaudit WEB se han utilizado herramientas y filtros standard existentes en todos los sistemas Unix (ver figura 5-16):

- **gzip**: Es el compresor de datos standard en sistemas Unix.

- **gunzip**: Es el descompresor de datos generados por gzip. Recordamos que IPaudit WEB almacena los datos obtenidos en formato comprimido (gzip).

- **zcat**: Es una utilidad del sistema que nos permite examinar ficheros comprimidos con gzip redireccionando su salida descomprimida hacia la salida standard (*standard output*).

- **awk**: Es una utilidad para el proceso de ficheros de texto. Este programa tiene un lenguaje de programación que permite la aplicación de complejos filtros a los datos.

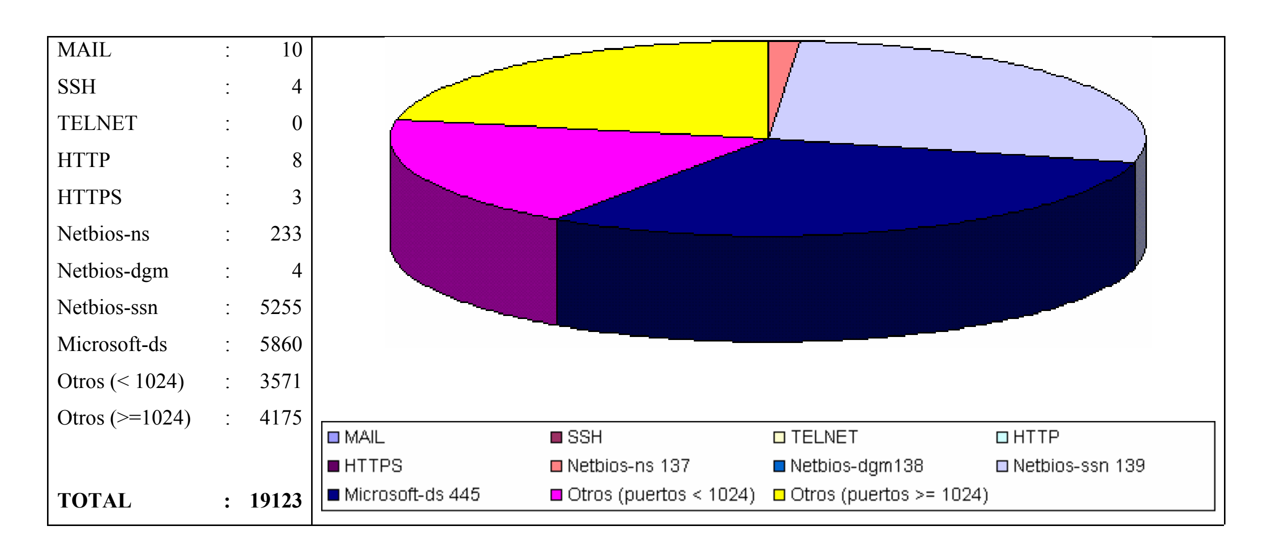
- Ejemplo de filtrado manual de los datos del día 21-08-2003:

```
/home/ipaudit/data/30min/# zcat 2003-08-21*
| awk '{ if (substr($2,1,8) != "192.168." && $12 == 2) { print }  }'
| awk '{ if ($4 >= 1024) { print} } '
| wc -l
```

FIG. 5-16: Ejemplo de filtrado fino de datos.

5.4.1 Día 21 de Agosto

El día 21 de agosto de 2003 se registraron un total de 19.123 paquetes enviados a nuestra red local. El desglose básico del tráfico clasificado por los principales servicios a los que hace referencia podemos observarlo en la figura 5-17.



MAIL	:	10
SSH	:	4
TELNET	:	0
HTTP	:	8
HTTPS	:	3
Netbios-ns	:	233
Netbios-dgm	:	4
Netbios-ssn	:	5255
Microsoft-ds	:	5860
Otros (< 1024)	:	3571
Otros (>=1024)	:	4175
TOTAL	**:**	**19123**

FIG. 5-17: Clasificación del tráfico del día 21/08/2003 por servicio.

Podemos observar en el gráfico de distribución del tráfico por servicios cómo existe una enorme cantidad de datagramas que hacen referencia a los servicios *Netbios/Microsoft-ds*. Este gran porcentaje (casi el 60% del total) nos sorprendió enormemente ya que

durante el resto de la semana este patrón se siguió repitiendo. Después de examinar el tráfico más detalladamente y consultar distinta bibliografía [Alex00][WWW178] [WWW179][WWW181][WWW182] hemos encontrado las siguientes explicaciones:

- Los sistemas basados en Windows (95, 98, NT, 2000, XP…) implementan un protocolo de red denominado **Netbios** y que se sitúa por encima de la pila IP (*IP stack*). Este protocolo "escucha" en diferentes puertos del sistema (137,138…) para proporcionar los distintos servicios de compartición de archivos por red.

- El comportamiento por defecto de los sistemas Windows conectados a una red es el de anunciar su presencia (y por tanto datos como el nombre de la máquina, dominio al que pertenece, recursos compartidos que ofrece…) al resto de máquinas conectadas. Este anuncio se realiza indiscriminadamente a toda la red local y a todos los ordenadores conectados sin ningún mecanismo de seguridad o autenticación.

- Si estos puertos son accesibles desde fuera de la red local[ℜ] (debido a la inexistencia de mecanismos de seguridad en nuestra red o a su una mala configuración), cualquier petición recibida en estos puertos sería contestada automáticamente por el sistema. De esta forma, cualquier problema de seguridad (*bug*) en estos servicios permitiría a un eventual atacante tomar el control del sistema.

- Los sistemas operativos Windows son los más utilizados en el mundo, según el buscador Google [WWW183] en junio de 2003 el 92% de los navegadores utilizados para acceder a su buscador utilizaban este sistema operativo.

- Al igual que cualquier software los sistemas operativos Windows son propensos a presentar fallos de seguridad que son utilizados por *hackers* o *blackhats* con el objetivo de hacerse con el control de la máquina. El envío indiscriminado de peticiones a estos servicios por toda Internet con la esperanza de que llegue a un ordenador con Windows (la mayoría) explica la gran cantidad de peticones registradas.

[ℜ] Para más información ver el capítulo 1 dónde se explican los distintos protocolos de Internet así como su encapsulación.

De esta forma, todo el tráfico *Netbios/Microsoft-ds* recibido hace referencia a peticiones de información de estos servicios. Debido a esta característica y al gran volumen de tráfico que representan, no los comentamos con más detalle en los informes diarios.

En el análisis del día 21 también podemos observar la existencia de 10 intentos de conexión a los protocolos de MAIL.

192.168.000.002 149.083.020.007 6 **110** 54867 108 58 2 1 11:18:29.7280 11:18:29.8747 2 2
192.168.000.002 149.083.020.007 6 **110** 51827 108 58 2 1 11:18:37.0295 11:18:37.1500 2 2
192.168.000.002 149.083.020.007 6 **110** 56729 108 58 2 1 11:21:48.5914 11:21:48.6971 2 2
192.168.000.002 149.083.020.007 6 **110** 35268 108 58 2 1 11:23:40.9491 11:23:41.2797 2 2
192.168.000.002 149.083.020.007 6 **110** 56892 108 58 2 1 11:25:25.7887 11:25:25.9939 2 2

192.168.000.002 149.083.020.007 6 **143** 54867 108 58 2 1 11:18:29.6776 11:18:29.7769 2 2
192.168.000.002 149.083.020.007 6 **143** 51827 108 58 2 1 11:18:37.0442 11:18:37.1566 2 2
192.168.000.002 149.083.020.007 6 **143** 56729 108 58 2 1 11:22:03.8638 11:22:04.2172 2 2
192.168.000.002 149.083.020.007 6 **143** 35268 108 58 2 1 11:23:44.1623 11:23:44.4669 2 2
192.168.000.002 149.083.020.007 6 **143** 56892 108 58 2 1 11:25:31.5112 11:25:31.8420 2 2

Estos intentos de conexión revelan la intención de realizar una comprobación de los puertos (*port scanning*) existentes en nuestro sistema con el objetivo de detectar la existencia de los servicios de lectura de correo (POP3 e IMAP) ya que las peticiones provienen de la misma dirección IP en un corto lapso de tiempo y al observar los logs del sistema no se ha observado ningún otro comportamiento anormal.

Análogamente observamos que las peticiones a los servicios de SSH, HTTP y HTTPS provienen de la misma dirección IP y en el mismo período de tiempo, lo que nos permite concluir que desde la dirección IP 149.083.020.007 nos están sondeando el sistema para adivinar que servicios tenemos accesibles (por ejemplo con la herramienta NMAP).

192.168.000.002 149.083.020.007 6 **22** 51827 108 58 2 1 11:18:37.0474 11:18:37.1597 2 2
192.168.000.002 149.083.020.007 6 **22** 56729 108 58 2 1 11:22:01.6413 11:22:01.8319 2 2
192.168.000.002 149.083.020.007 6 **22** 35268 108 58 2 1 11:23:48.5159 11:23:48.7246 2 2
192.168.000.002 149.083.020.007 6 **22** 56892 108 58 2 1 11:25:29.4515 11:25:29.8100 2 2

192.168.000.002 149.083.020.007 6 **80** 51827 108 58 2 1 11:18:37.0375 11:18:37.1532 2 2
192.168.000.002 149.083.020.007 6 **80** 56729 108 58 2 1 11:21:58.2244 11:21:58.5682 2 2
192.168.000.002 149.083.020.007 6 **80** 35268 108 58 2 1 11:23:48.6333 11:23:48.9217 2 2
192.168.000.002 149.083.020.007 6 **80** 56892 108 58 2 1 11:25:27.4131 11:25:27.6054 2 2

Podemos observar también como recibimos algunas peticiones al servicio HTTP (puerto 80) desde otras direcciones IP. Como son pocas peticiones, aisladas y no se repiten, a priori podríamos concluir que probablemente no son más errores que ha cometido un usuario al teclear la dirección IP en su navegador.

192.168.000.002 081.096.155.177 6 **80** 4493 481 519 6 4 05:03:48.5840 05:04:24.5418 2 2
192.168.000.002 199.035.016.165 6 **80** 5632 108 58 2 1 06:48:57.7993 06:48:58.2571 2 2
192.168.000.002 080.004.006.139 6 **80** 3310 337 524 5 4 08:35:25.7793 08:35:26.5009 2 2
192.168.000.002 146.145.025.067 6 **80** 2146 318 531 5 5 22:42:10.8798 22:42:11.4060 2 2

Sin embargo al observar los logs del servidor WWW para conocer los comandos que se han ejecutado desde estas direcciones IP podemos observar con cierta sorpresa que estamos siendo víctimas de distintos ataques [WWW185].

81.96.155.177 -- [21/Aug/2003:05:03:48 +0200] "OPTIONS / HTTP/1.1" 200 –
146.145.25.67 -- [21/Aug/2003:22:42:11 +0200] "HEAD / HTTP/1.0" 200 0

La bibliografía consultada [PU02][WWW186][WWW187][WWW194] nos indica que muchas aplicaciones de *hackers* o *blackhats* comprueban la posibilidad de utilizar los servidores WWW como sistemas *proxy*[ℜ]. La idea que se esconde tras estas peticiones es la de buscar el anonimato del atacante de forma que pueda utilizar el servidor WWW-*proxy* como lanzadera de peticiones a otros servidores de forma indirecta.

80.4.6.139 -- [21/Aug/2003:08:35:26+0200] "GET/scripts/..%255c%255c../
winnt/system32/cmd.exe?/c+dir"

En esta otra petición el atacante de la dirección 80.4.6.139 busca un sistema Windows que tenga un servidor WWW mal configurado o anticuado (que no se hayan aplicados los distintos parches de seguridad que van saliendo). Su objetivo es la comprobación de la posibilidad de ejecutar comandos arbitrarios en el servidor atacado. En este caso para realizar la comprobación simplemente prueba de ejecutar un simple comando '*dir*'.

En el caso de los accesos HTTPS vemos que hemos recibidos tres peticiones de conexión. Sin embargo como este servicio no está en funcionamiento no puede causarnos mas problemas que el de recibir tráfico no solicitado.

[ℜ]El sistema de *proxy* más famoso actualmente es el SQUID [WWW91].

192.168.000.002 149.083.020.007 6 **443** 56729 54 54 1 1 11:21:58.8044 11:21:58.8044 2 1
192.168.000.002 149.083.020.007 6 **443** 35268 54 54 1 1 11:23:49.3747 11:23:49.3747 2 1
192.168.000.002 149.083.020.007 6 **443** 56892 54 54 1 1 11:25:21.4252 11:25:21.4252 2 1

A continuación realizaremos otro desglose del tráfico de la red local recibido según el tipo de protocolo (TCP, ICMP, UDP y Otros) al que hacen referencia (ver figura 5-18).

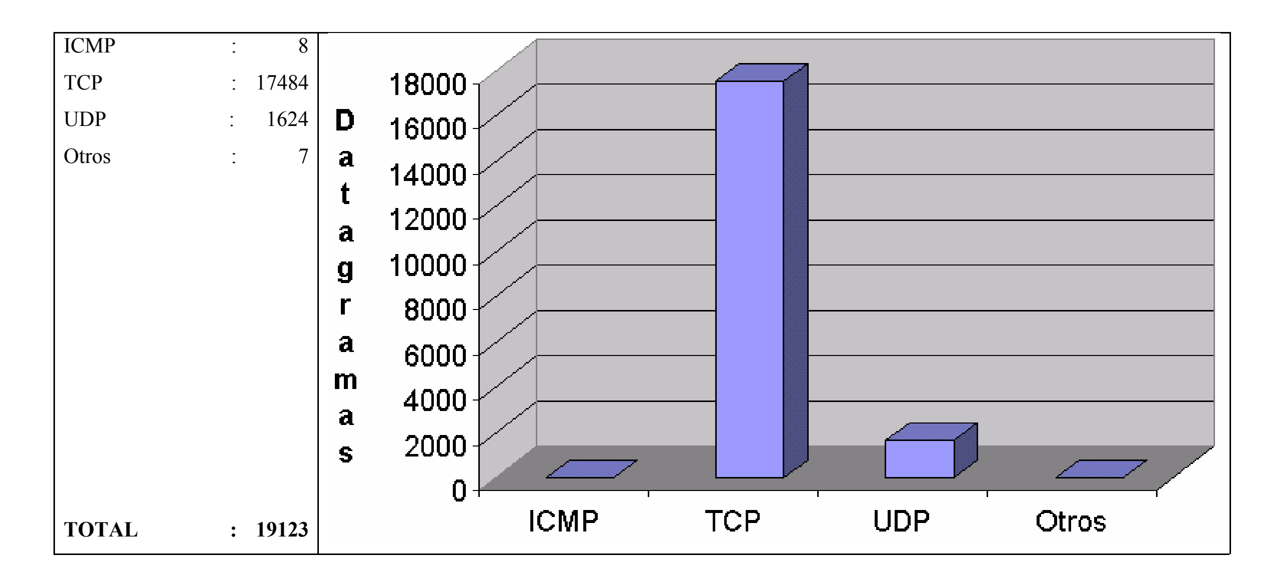


FIG. 5-18: distribución del tráfico del día 21/08/2003 por protocolo.

Lo que realmente nos llamó la atención sobre el tipo de tráfico recibido fueron los siete datagramas clasificados en la categoría de otros:

127.000.000.001 127.000.000.001 **4** 0 0 54 0 1 0 18:34:59.1733 18:34:59.1733 2 2

Este datagrama tiene dirección 127.0.0.1 (dirección local o *loopback* de cualquier ordenador) como origen y destino. Además se identifica como protocolo 4 (inexistente en las especificaciones [Ric98-1]) y se dirige del puerto 0 del ordenador origen al puerto 0 del ordenador destino (los puertos van del 1 al 65535).

Probablemente se trate de un paquete mal construido por algún software de los que tenemos instalado o un *bug* del sistema operativo. En cualquier caso no se ha vuelto a repetir durante el experimento.

000.011.000.000 193.197.192.168 **4** 0 0 7868592 0 3816 0 18:57:42.5740 18:57:53.3571 2 2

Este otro datagrama es también bastante extraño, ya que si bien podemos observar que se identifica perfectamente por la dirección IP de origen (193.197.192.168) no debería haber llegado nunca a nuestra red. Además podemos observar como pertenece a un protocolo inexistente (4) y sus puertos de origen y destino son inválidos.

255.000.000.000 255.255.255.255 **0** 0 0 14 0 1 0 18:57:53.0193 18:57:53.0193 2 2
000.000.000.000 000.000.000.000 **0** 0 0 112 0 8 0 18:57:53.0213 18:58:16.5709 2 2
000.000.000.000 255.255.255.000 **0** 0 0 14 0 1 0 18:57:53.0217 18:57:53.0217 2 2
000.000.000.000 118.118.118.118 **0** 0 0 14 0 1 0 18:57:53.0295 18:57:53.0295 2 2
000.000.000.000 118.118.118.000 **0** 0 0 14 0 1 0 18:57:53.2009 18:57:53.2009 2 2

Análogamente, estos otros datagramas también están construidos de forma anómala. Sin embargo, el hecho de que se registren tan cercanos en el tiempo (entre las 18:57 y las 18:58) y no hayan vuelto a producirse durante la semana, nos hace creer que son paquetes erróneos creados por el *router* o algún software instalado localmente (como por ejemplo el de captura del tráfico).

En cuanto a los datagramas ICMP recibidos son datagramas destinados al puerto de destino 0. Este número de puerto es incorrecto ya que las especificaciones formales de IETF e IANA señalan el rango válido del puerto 1 al 65535.

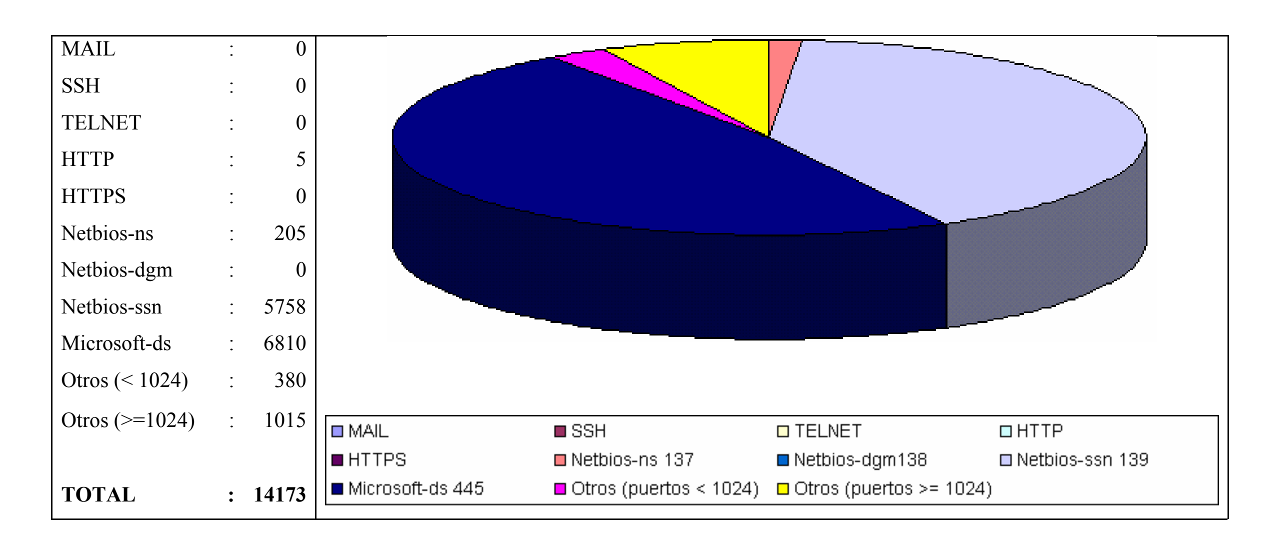
192.168.000.002 064.014.070.082 **1** 0 769 70 0 1 0 00:00:13.3872 00:00:13.3872 2 2
192.168.000.002 064.014.070.082 **1** 0 769 70 0 1 0 00:30:08.9668 00:30:08.9668 2 2
192.168.000.002 068.213.215.242 **1** 0 771 299 0 1 0 01:06:53.2221 01:06:53.2221 2 2
192.168.000.002 212.033.064.003 **1** 0 771 124 0 1 0 01:30:14.9810 01:30:14.9810 2 2
192.168.000.002 203.036.248.001 **1** 0 771 120 0 1 0 03:00:04.7731 03:00:04.7731 2 2
192.168.000.011 149.083.020.012 **1** 0 0 148 0 2 0 16:22:55.7337 16:23:04.2181 2 2
192.168.000.002 200.164.059.251 **1** 0 769 70 0 1 0 21:45:28.1387 21:45:28.1387 2 2
192.168.000.002 212.033.064.003 **1** 0 771 124 0 1 0 22:01:11.4647 22:01:11.4647 2 2

Probablemente estos paquetes sean ejemplos de intentos de adivinación del sistema operativo existente. La herramienta NMAP (como muchas otras) permite la posibilidad de descubrir el sistema operativo remoto enviando datagramas específicamente construidos (*OS fingerprinting*). Las reacciones del sistema a peticiones al puerto 0 del protocolo ICMP es una de las técnicas más utilizadas.

El resto del tráfico TCP y UDP registrado hace referencia a las peticiones *Netbios/Microsoft-ds* por lo que no serán estudiadas con más profundidad en los informes diarios.

5.4.2 Día 22 de Agosto

El día 22 de agosto de 2003 se registraron un total de 14.173 paquetes enviados a nuestra red local. El desglose básico del tráfico clasificado por los principales servicios a los que hace referencia podemos observarlo en la figura 5-19.



MAIL	:	0
SSH	:	0
TELNET	:	0
HTTP	:	5
HTTPS	:	0
Netbios-ns	:	205
Netbios-dgm	:	0
Netbios-ssn	:	5758
Microsoft-ds	:	6810
Otros (< 1024)	:	380
Otros (>=1024)	:	1015
TOTAL	**:**	**14173**

FIG. 5-19: Clasificación del tráfico del día 22/08/2003 por servicio.

Podemos observar nuevamente como el tráfico de los servicios *Netbios/Microsoft-ds* es el gran dominante. También podemos observar la existencia de cinco peticiones realizadas al servicio HTTP:

192.168.000.002 081.096.155.177 6 **80** 4320 427 519 5 4 01:36:19.3746 01:36:36.3773 2 2
192.168.000.002 064.222.171.148 6 **80** 4752 483 458 6 4 01:37:48.3108 01:38:08.2907 2 2
192.168.000.002 195.249.206.242 6 **80** 2852 314 527 5 5 07:09:08.9599 07:09:09.2694 2 2
192.168.000.002 066.183.188.131 6 **80** 1090 300 446 5 5 07:35:45.6170 07:35:46.3324 2 2
192.168.000.002 195.227.096.181 6 **80** 53470 253 503 4 4 23:39:22.6346 23:39:22.9817 2 2

Una vez examinados los logs podemos observar que las peticiones realizadas al servidor WWW se tratan de ataques similares a los registrados el día 21 de Agosto. Vemos que algunas peticiones sondean la existencia de un sistema *proxy* y también ataques a la espera de encontrar un sistema Windows con un servidor Microsoft IIS vulnerable:

```
81.96.155.177    -- [22/Aug/2003:01:36:19 +0200] "OPTIONS / HTTP/1.1" 200 –
195.249.206.242 -- [22/Aug/2003:07:09:09 +0200] "HEAD / HTTP/1.0" 200 0
66.183.188.131   -- [22/Aug/2003:07:35:45 +0200] "OPTIONS * HTTP/1.0" 200 –

195.227.96.181   -- [22/Aug/2003:23:39:22 +0200] "GET /scripts/nsiislog.dll" 404 –
```

Además, al observar el log del servidor WWW vemos una petición nada usual ya que intenta obtener la página principal de http://www.yahoo.com.

```
64.222.171.148   -- [22/Aug/2003:01:37:49 +0200] "GET http://www.yahoo.com/ HTTP/1.1"
```

La idea del atacante es comprobar si existe un servidor WWW mal configurado que permita las referencias a sistemas externos. De esta forma podrían utilizarlo como lanzadera contra otros sitios de Internet.

En la figura 5-20 podemos observar el desglose del tráfico recibido según el tipo de protocolo (TCP, ICMP, UDP y Otros) al que hacen referencia.

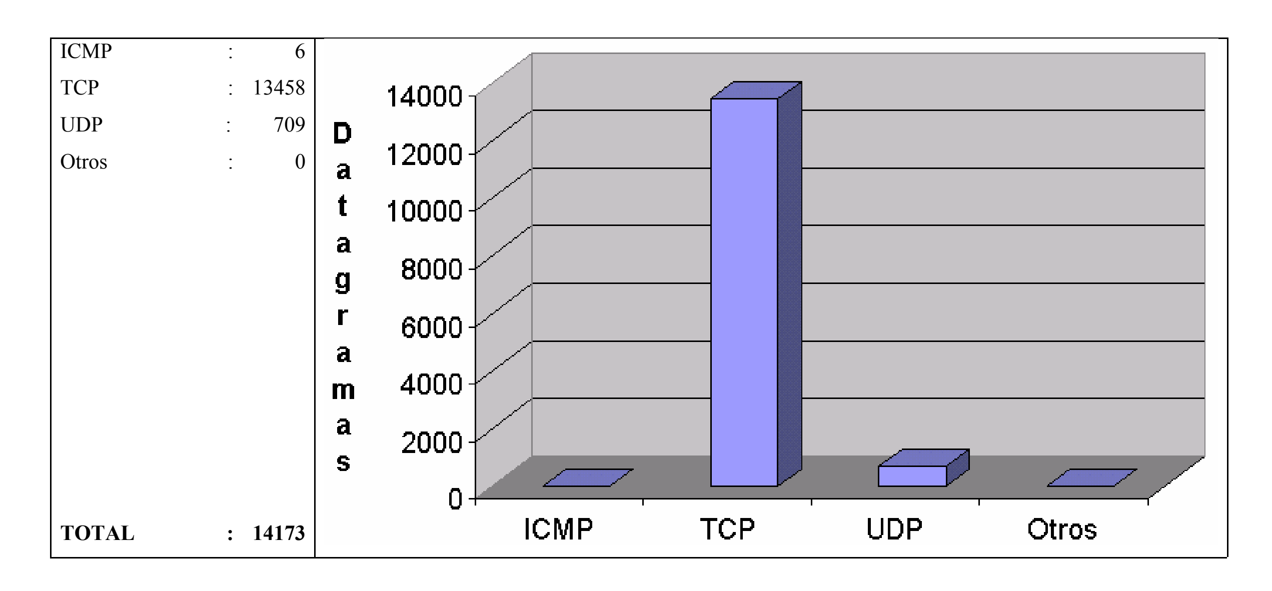


FIG. 5-20: distribución del tráfico del día 22/08/2003 por protocolo.

Observamos que durante el día se han recibido seis datagramas ICMP que presentan dos características comunes inquietantes:

192.168.000.002 193.205.245.008 **1** 0 771 70 0 1 0 01:00:48.9558 01:00:48.9558 2 2
192.168.000.002 212.033.064.003 **1** 0 771 124 0 1 0 03:30:08.6266 03:30:08.6266 2 2
192.168.000.002 200.049.165.248 **1** 0 771 299 0 1 0 10:43:14.0115 10:43:14.0115 2 2
192.168.000.002 080.218.038.007 **1** 0 771 299 0 1 0 17:20:40.2051 17:20:40.2051 2 2
192.168.000.002 209.102.127.086 **1** 0 771 299 0 1 0 23:20:30.3301 23:20:30.3301 2 2
192.168.000.002 200.045.217.105 **1** 0 771 299 0 1 0 23:51:23.0115 23:51:23.0115 2 2

1. Son datagramas ICMP que van destinados al puerto 0 de nuestro servidor local.

2. Todas provienen del mismo puerto de origen (771).

Al igual que parte del tráfico ICMP del día 21, nos encontramos con algunas peticiones que provienen de un mismo puerto [WWW188]. Aunque y que no hemos encontrado en la bibliografía ninguna referencia a posibles ataques, virus o gusanos que presenten estas características, estamos convencidos de que proviene de alguna herramienta específica poco común o no muy extendida actualmente (ya hemos recibido pocas peticiones).

La probabilidad de recibir la misma petición desde dos o más direcciones IP distintas con el mismo puerto de origen es prácticamente nula.

5.4.3 Día 23 de Agosto

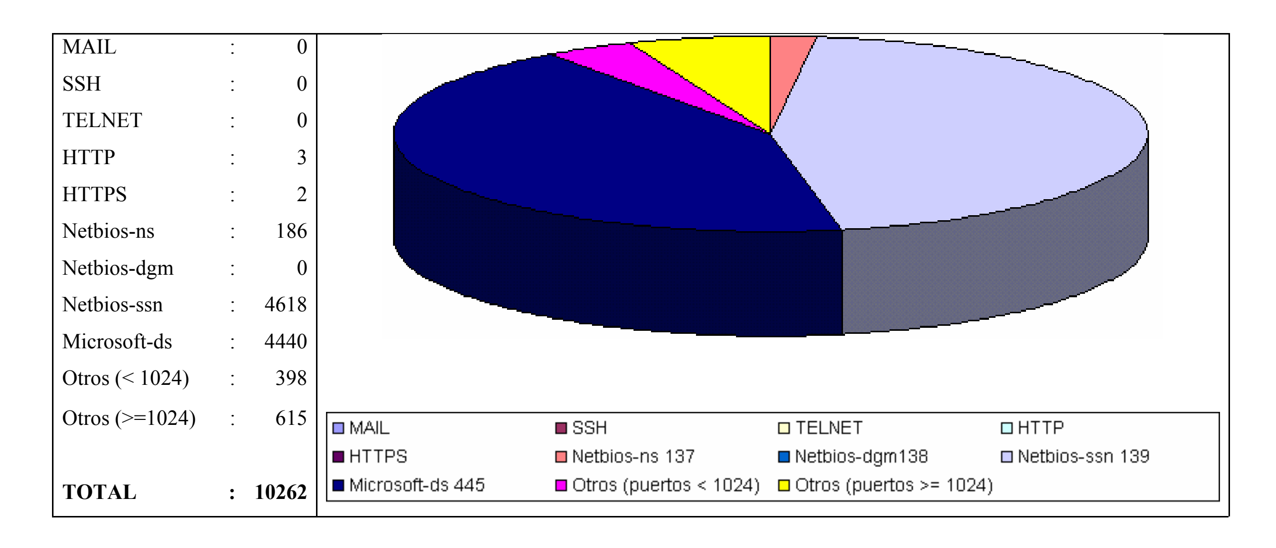
El día 23 de agosto de 2003 se registraron un total de 10.262 paquetes enviados a nuestra red local. El desglose básico del tráfico clasificado por los principales servicios a los que hace referencia podemos observarlo en la figura 5-21.

Una vez más, el tráfico de los servicios *Netbios/Microsoft-ds* es el más abundante de los registrados. De todo el tráfico recibido destacamos las tres peticiones al servicio HTTP recibidas.

192.168.000.002 217.082.197.149 6 **80** 4900 4154 773 6 6 04:57:33.7074 04:57:34.4797 2 2
192.168.000.002 218.005.066.254 6 **80** 1148 4150 769 6 6 16:43:18.6133 16:43:22.8677 2 2
192.168.000.002 193.115.134.132 6 **80** 4406 307 503 5 4 17:39:39.4002 17:39:39.7001 2 2

Al observar los logs del servidor WWW y ver las peticiones que se han realizado podemos concluir que hemos recibido un ataque en toda regla del virus CodeRed II [WWW184]. Este virus busca servidores WWW de Microsoft para propagarse (*IIS, Internet Information Server*).

217.82.197.149 -- [23/Aug/2003:04:57:34 +0200] "GET /default.ida?XXX……..
218.5.66.254 -- [23/Aug/2003:16:43:21 +0200] "GET /default.ida?XXX………
193.115.134.132 -- [23/Aug/2003:17:39:39 +0200] "GET /scripts/nsiislog.dll" 404 –



MAIL	:	0
SSH	:	0
TELNET	:	0
HTTP	:	3
HTTPS	:	2
Netbios-ns	:	186
Netbios-dgm	:	0
Netbios-ssn	:	4618
Microsoft-ds	:	4440
Otros (< 1024)	:	398
Otros (>=1024)	:	615
TOTAL	**:**	**10262**

FIG. 5-21: Clasificación del tráfico del día 23/08/2003 por servicio.

En el caso de los accesos al servicio HTTPS vemos que hemos recibidos dos peticiones de conexión. Sin embargo como este servicio no está en funcionamiento no puede causarnos mas problemas que el de recibir tráfico no solicitado.

192.168.000.002 207.044.130.095 6 **443** 34775 74 54 1 1 09:07:30.8778 09:07:30.8780 2 1
192.168.000.002 211.110.012.055 6 **443** 38958 74 54 1 1 09:51:56.5157 09:51:56.5159 2 1

A continuación podemos observar el desglose del tráfico según el tipo de protocolo (TCP, ICMP, UDP y Otros) al que hacen referencia del día 23 en la figura 5-22.

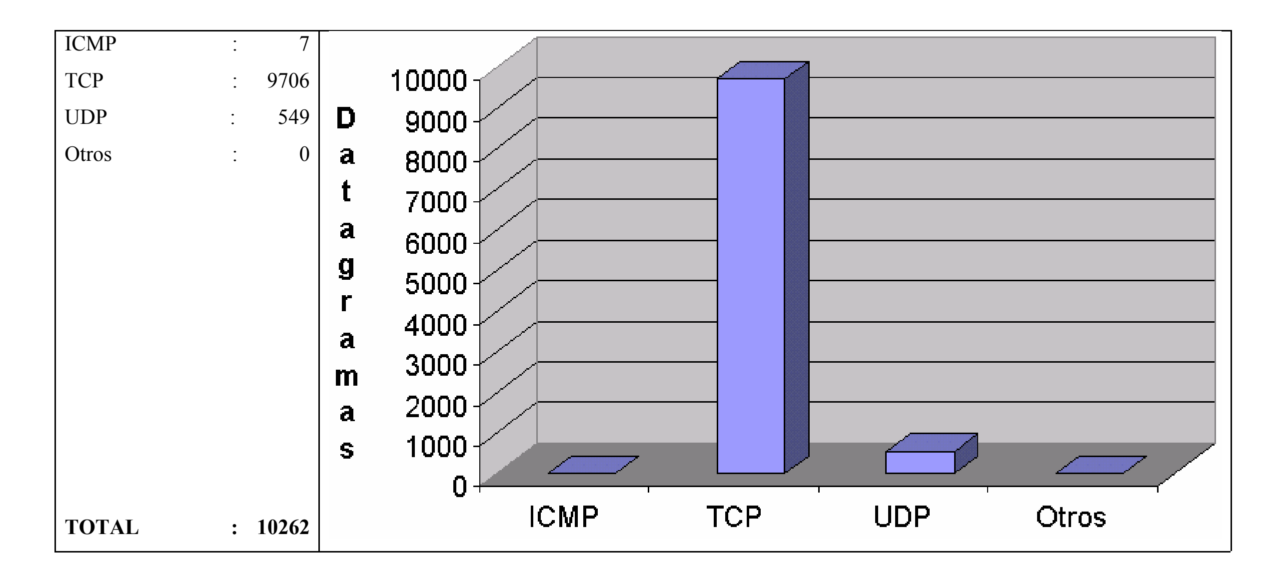


FIG. 5-22: distribución del tráfico del día 23/08/2003 por protocolo.

El día 23 volvemos a observar algunas peticiones ICMP extrañas desde varias direcciones distintas con dos puertos concretos de origen (771 y 781) y con destino el puerto local 0.

192.168.000.002 148.081.027.094 **1** 0 781 70 0 1 0 01:00:49.5464 01:00:49.5464 2 2
192.168.000.002 063.152.127.062 **1** 0 781 350 0 5 0 01:51:18.6785 01:52:49.6403 2 2
192.168.000.002 212.033.064.003 **1** 0 771 123 0 1 0 02:00:14.4856 02:00:14.4856 2 2
192.168.000.002 144.232.018.002 **1** 0 781 70 0 1 0 06:30:14.6294 06:30:14.6294 2 2
192.168.000.002 212.074.093.014 **1** 0 781 70 0 1 0 08:30:25.7445 08:30:25.7445 2 2
192.168.000.002 212.033.064.003 **1** 0 771 124 0 1 0 20:00:14.3016 20:00:14.3016 2 2
192.168.000.002 063.237.032.218 **1** 0 781 140 0 2 0 23:39:01.8081 23:39:04.8241 2 2

5.4.4 Día 24 de Agosto

El día 24 de agosto de 2003 se registraron un total de 13.638 paquetes enviados a nuestra red local. El desglose básico del tráfico clasificado por los principales servicios a los que hace referencia podemos observarlo en la figura 5-23.

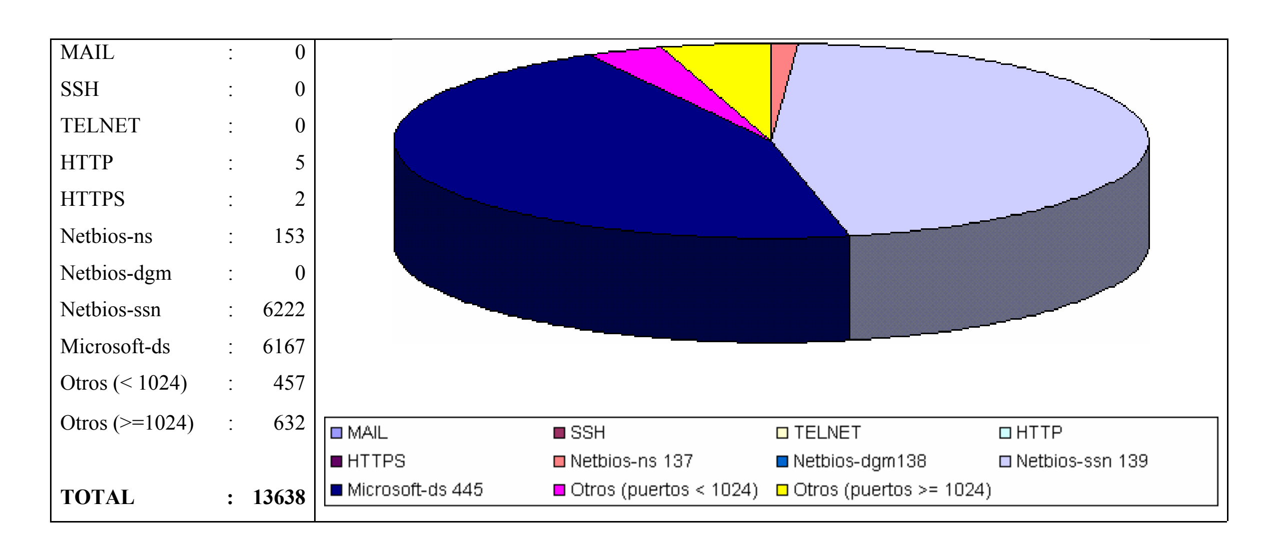
Como en días anteriores, el tráfico de los servicios *Netbios/Microsoft-ds* es el más abundante de los registrados. También podemos destacar las cinco peticiones al servicio HTTP recibidas:

192.168.000.002 217.162.137.110 6 **80** 1251 4204 1314 7 8 02:47:31.3763 02:47:38.6239 2 1
192.168.000.002 153.109.141.031 6 **80** 1198 311 819 5 4 05:02:45.0445 05:02:45.3621 2 2
192.168.000.002 065.192.064.082 6 **80** 4218 318 531 5 5 09:45:48.9481 09:45:49.4206 2 2
192.168.000.002 217.058.137.120 6 **80** 3632 258 666 4 4 21:19:56.9372 21:19:57.2922 2 2
192.168.000.002 149.099.032.020 6 **80** 3635 309 886 5 4 22:02:02.1511 22:02:02.8126 2 2

Una vez más analizamos los logs del servidor WWW para intentar descubrir el objetivo de estas peticiones. Al igual que en los días anteriores recibimos intentos de ataques a sistemas Windows y servidores IIS así como pruebas de las capacidades *proxy* que podría presentar nuestro servidor.

```
217.58.137.120   -- [24/Aug/2003:21:19:57 +0200] "GET /msadc/msadcs.dll HTTP/1.0" 404 275
217.162.137.110 -- [24/Aug/2003:02:47:35 +0200] "GET /default.ida?XXX%u9090%u6858%ucbd3
                                                 %u7801%u9090%u6858%ucbd3%u7801%u9090%u
                                                 6858%ucbd3%u7801%u9090%u9090%u8190%u00
                                                 c3%u0003%u8b00%u531b%u53ff%u0078%u0000
                                                 %u00=a  HTTP/1.0" 404 270

153.109.141.31 -- [24/Aug/2003:05:02:45 +0200] "CONNECT 1.3.3.7:1337 HTTP/1.0" 405 296
65.192.64.82    -- [24/Aug/2003:09:45:49 +0200] "HEAD / HTTP/1.0" 200 0
149.99.32.20    -- [24/Aug/2003:22:02:02 +0200] "SEARCH / HTTP/1.1" 501 331
```



MAIL	:	0
SSH	:	0
TELNET	:	0
HTTP	:	5
HTTPS	:	2
Netbios-ns	:	153
Netbios-dgm	:	0
Netbios-ssn	:	6222
Microsoft-ds	:	6167
Otros (< 1024)	:	457
Otros (>=1024)	:	632
TOTAL	**:**	**13638**

FIG. 5-23: Clasificación del tráfico del día 24/08/2003 por servicio.

A continuación en la figura 5-24 realizaremos un el desglose de todo el tráfico recibido durante el día según el tipo de protocolo (TCP, ICMP, UDP y Otros) al que hacen referencia.

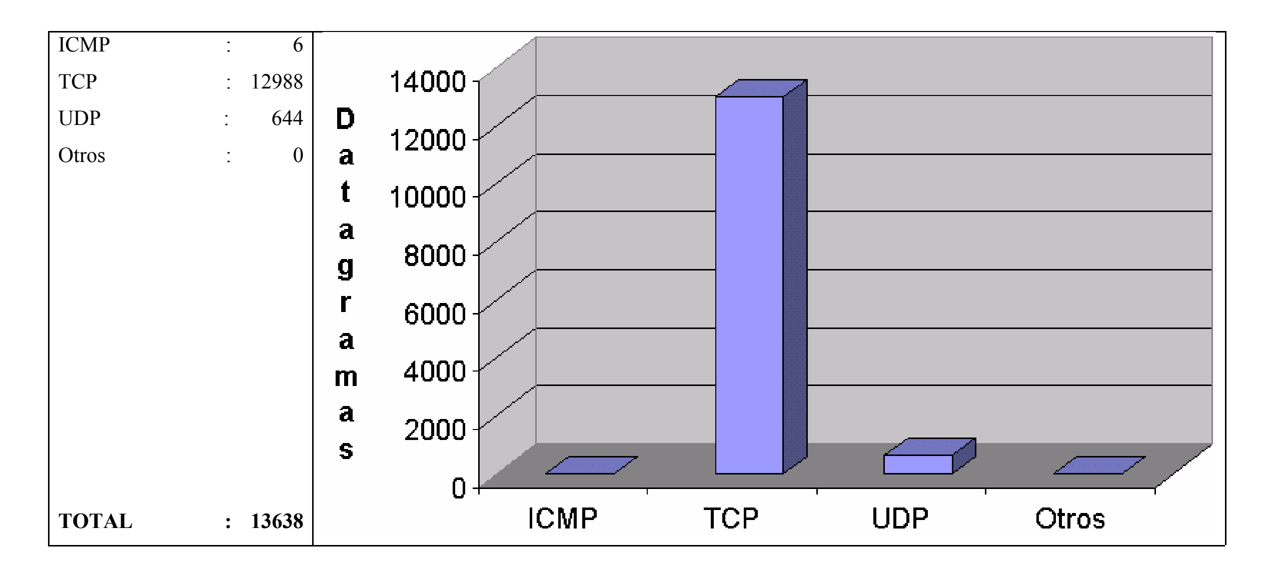


FIG. 5-24: distribución del tráfico del día 24/08/2003 por protocolo.

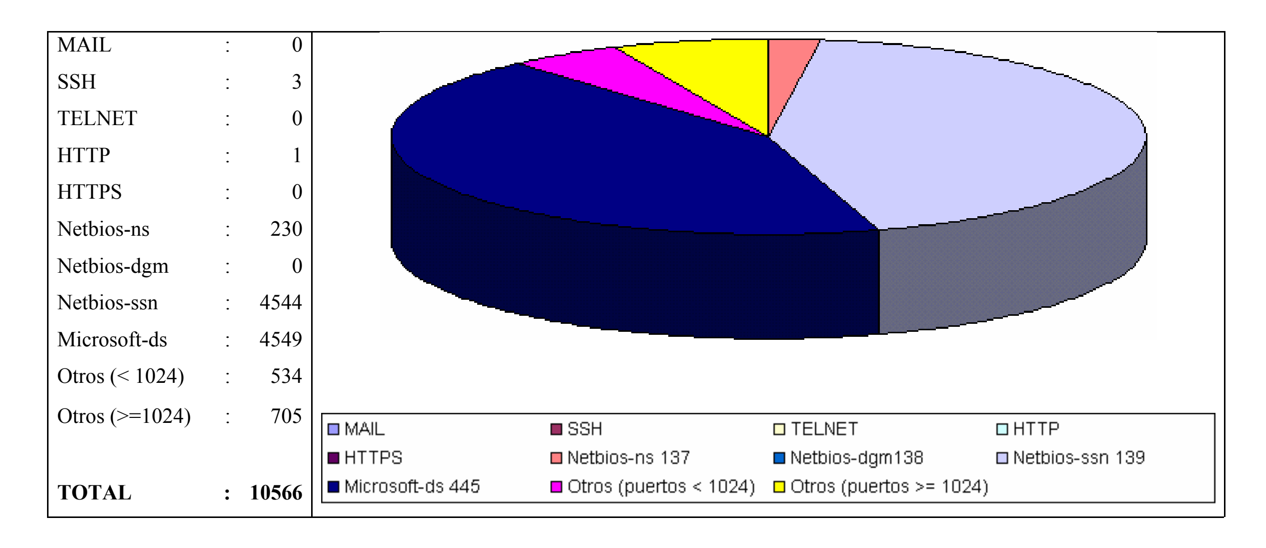
Al igual que en los días anteriores registramos algunas peticiones ICMP extrañas desde varias direcciones distintas con el puerto de origen 771 y con destino el puerto local 0.

192.168.000.002 198.005.241.058 **1** 0 771 234 0 2 0 04:01:06.8198 04:01:06.8295 2 2
192.168.000.002 212.033.064.003 **1** 0 771 124 0 1 0 05:30:20.7972 05:30:20.7972 2 2
192.168.000.002 217.096.223.189 **1** 0 771 299 0 1 0 05:57:56.3417 05:57:56.3417 2 2
192.168.000.002 211.158.093.040 **1** 0 771 299 0 1 0 11:21:30.9523 11:21:30.9523 2 2
192.168.000.002 212.033.064.003 **1** 0 771 124 0 1 0 13:30:14.9085 13:30:14.9085 2 2
192.168.000.002 203.148.128.017 **1** 0 771 299 0 1 0 23:12:22.1506 23:12:22.1506 2 2

5.4.5 Día 25 de Agosto

El día 25 de agosto de 2003 se registraron un total de 10.566 paquetes enviados a nuestra red local. El desglose básico del tráfico clasificado por los principales servicios a los que hace referencia podemos observarlo en la figura 5-25.

Como en días anteriores, el tráfico de los servicios *Netbios/Microsoft-ds* es el más abundante de los registrados. Podemos observar la existencia de varias peticiones externas "simultáneas" al servicio SSH de nuestra red local desde una dirección IP idéntica:



MAIL	:	0
SSH	:	3
TELNET	:	0
HTTP	:	1
HTTPS	:	0
Netbios-ns	:	230
Netbios-dgm	:	0
Netbios-ssn	:	4544
Microsoft-ds	:	4549
Otros (< 1024)	:	534
Otros (>=1024)	:	705
TOTAL	**:**	**10566**

FIG. 5-25: Clasificación del tráfico del día 25/08/2003 por servicio.

192.168.000.002 213.201.169.102 6 22 29311 350 183 5 2 19:50:38.5318 19:50:42.0202 2 2
192.168.000.002 213.201.169.102 6 22 29455 2518 3371 18 20 19:50:41.9192 19:51:58.1431 2 2
192.168.000.002 213.201.169.102 6 22 29542 2584 3297 19 19 19:50:45.1521 19:51:59.7611 2 2

Tras analizar detenidamente el flujo de las conexiones (ver figura 5-26) observamos que las peticiones realizan conexiones (SYN) y desconexiones (RST) rápidamente sin realizar ningún intercambio de datos (Len=0) por lo que probablemente el atacante buscaba un servidor SSH vulnerable a este ataque.

También podemos observar como se ha recibido una única petición al servicio HTTP durante todo el día.

192.168.000.002 209.157.068.242 6 **80** 1380 307 503 5 4 14:47:55.3717 14:47:56.0154 2 2

209.157.68.242 - - [25/Aug/2003:14:47:55 +0200] "GET /scripts/nsiislog.dll" 404 -

Una vez más el análisis de los logs del servidor nos revela que se trataba de un ataque en busca de un sistema operativo Windows mal configurado o no actualizado con IIS como servidor WWW.

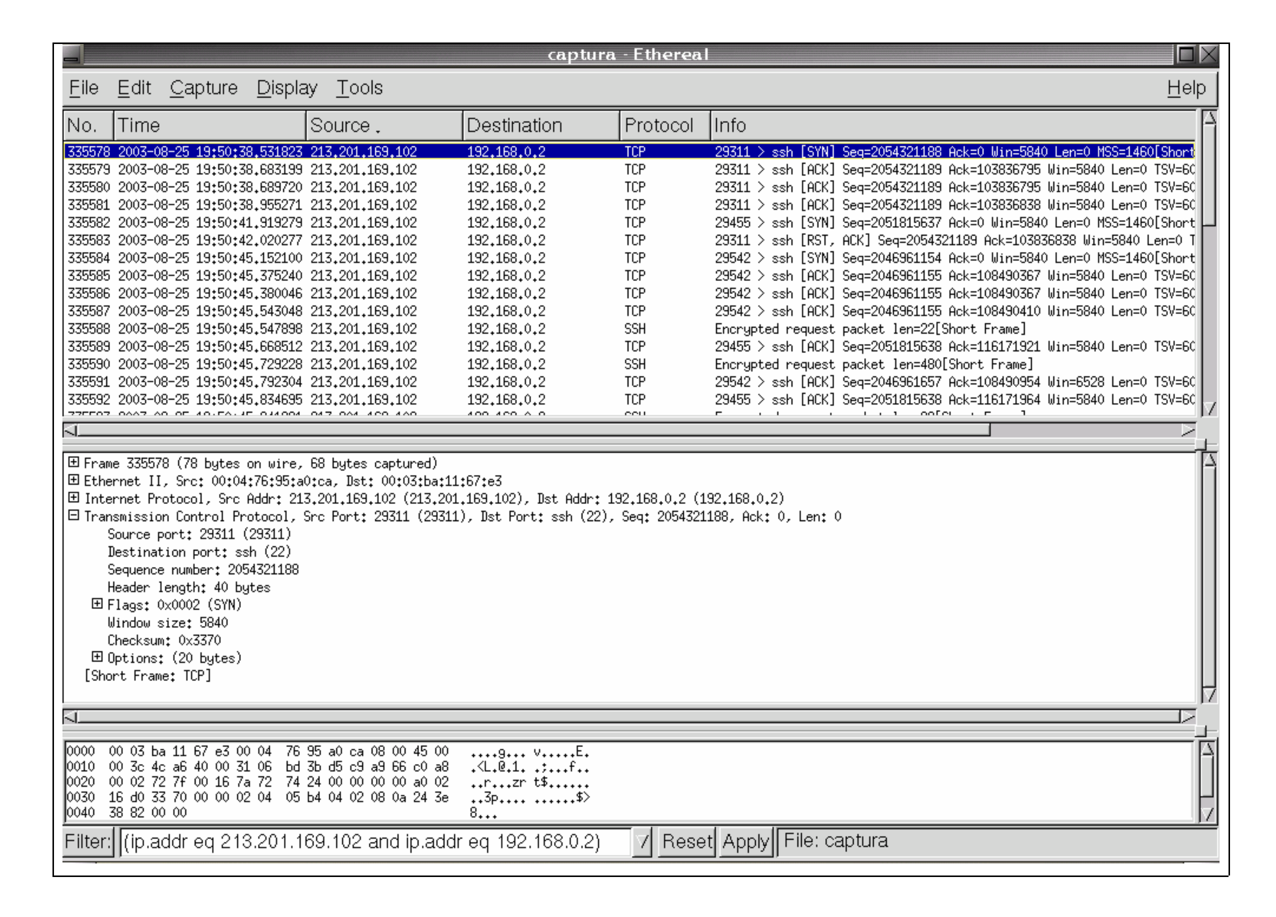


FIG. 5-26: Análisis del tráfico SSH recibido el día 25/08/2003.

A continuación podemos observar el desglose del tráfico del día 25 según el tipo de protocolo (TCP, ICMP, UDP y Otros) al que hacen referencia en la figura 5-27.

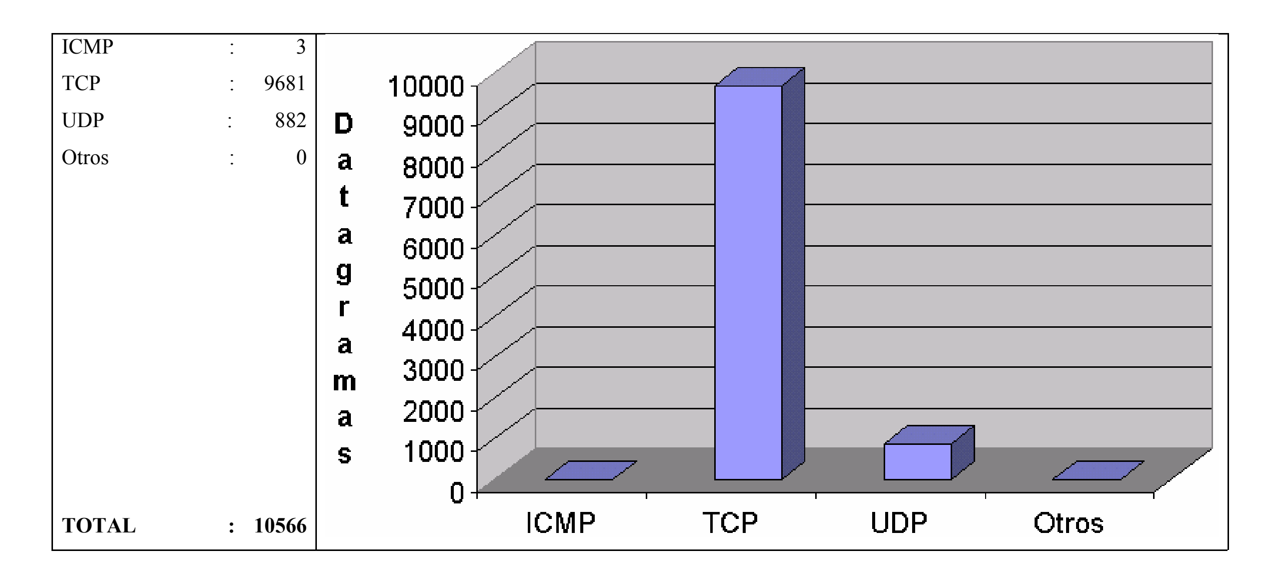


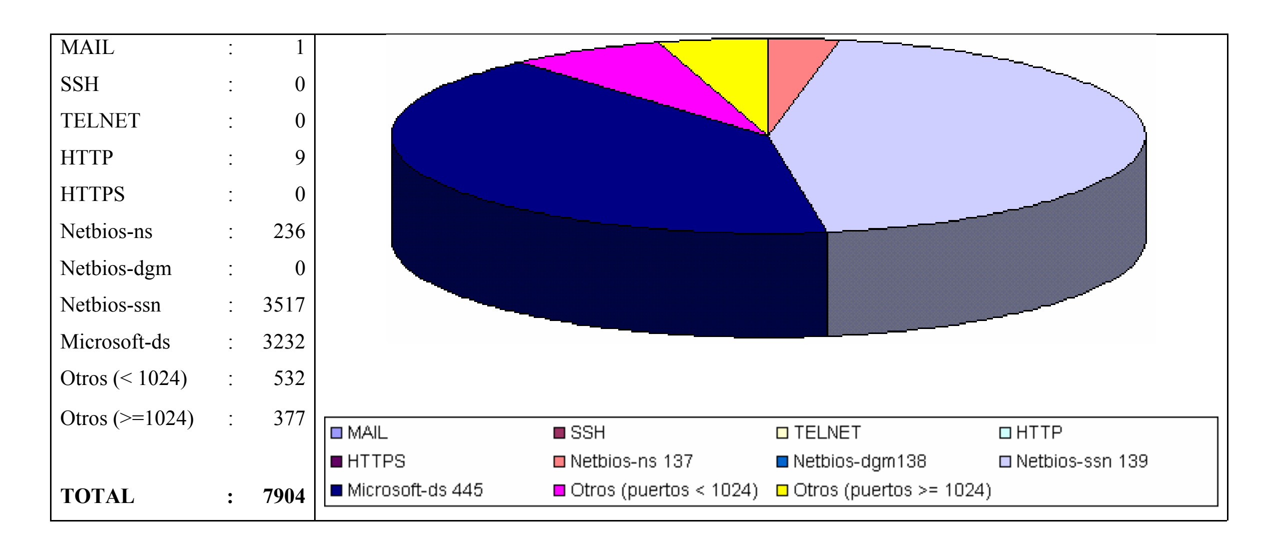
FIG. 5-27: distribución del tráfico del día 25/08/2003 por protocolo.

Como en los días precedentes registramos algunas peticiones ICMP extrañas desde varias direcciones distintas con el puerto de origen 769, 771 o 781 y con destino el puerto 0.

192.168.000.002 196.025.249.070 **1** 0 781 70 0 1 0 13:00:17.5611 13:00:17.5611 2 2
192.168.000.002 080.058.036.083 **1** 0 769 70 0 1 0 14:30:40.7625 14:30:40.7625 2 2
192.168.000.002 212.033.064.003 **1** 0 771 124 0 1 0 17:30:19.8907 17:30:19.8907 2 2

5.4.6 Día 26 de Agosto

El día 26 de agosto de 2003 se registraron un total de 7.904 paquetes enviados a nuestra red local. El desglose básico del tráfico clasificado por los principales servicios a los que hace referencia podemos observarlo en la figura 5-28.



MAIL	:	1
SSH	:	0
TELNET	:	0
HTTP	:	9
HTTPS	:	0
Netbios-ns	:	236
Netbios-dgm	:	0
Netbios-ssn	:	3517
Microsoft-ds	:	3232
Otros (< 1024)	:	532
Otros (>=1024)	:	377
TOTAL	**:**	**7904**

FIG. 5-28: Clasificación del tráfico del día 26/08/2003 por servicio.

Como en días anteriores, el tráfico de los servicios *Netbios/Microsoft-ds* es el más abundante de los registrados. Se ha recibido una única petición de conexión al servicio de MAIL:

192.168.000.002 211.213.123.254 6 **25** 1969 601 624 9 9 22:18:26.2661 22:18:29.8132 2 1

Tras realizar un análisis exhaustivo de los logs del sistema, comprobamos con estupor (ver figura 5-29) cómo el objetivo de esta comunicación es el de comprobar si nuestro sistema de MAIL permite el envío de correos electrónicos a cualquier dirección sin restricciones (*open relaying*). Este tipo de búsquedas persiguen encontrar servidores SMTP mal configurados de forma que puedan ser aprovechados para el envío de correo basura (*spam*).

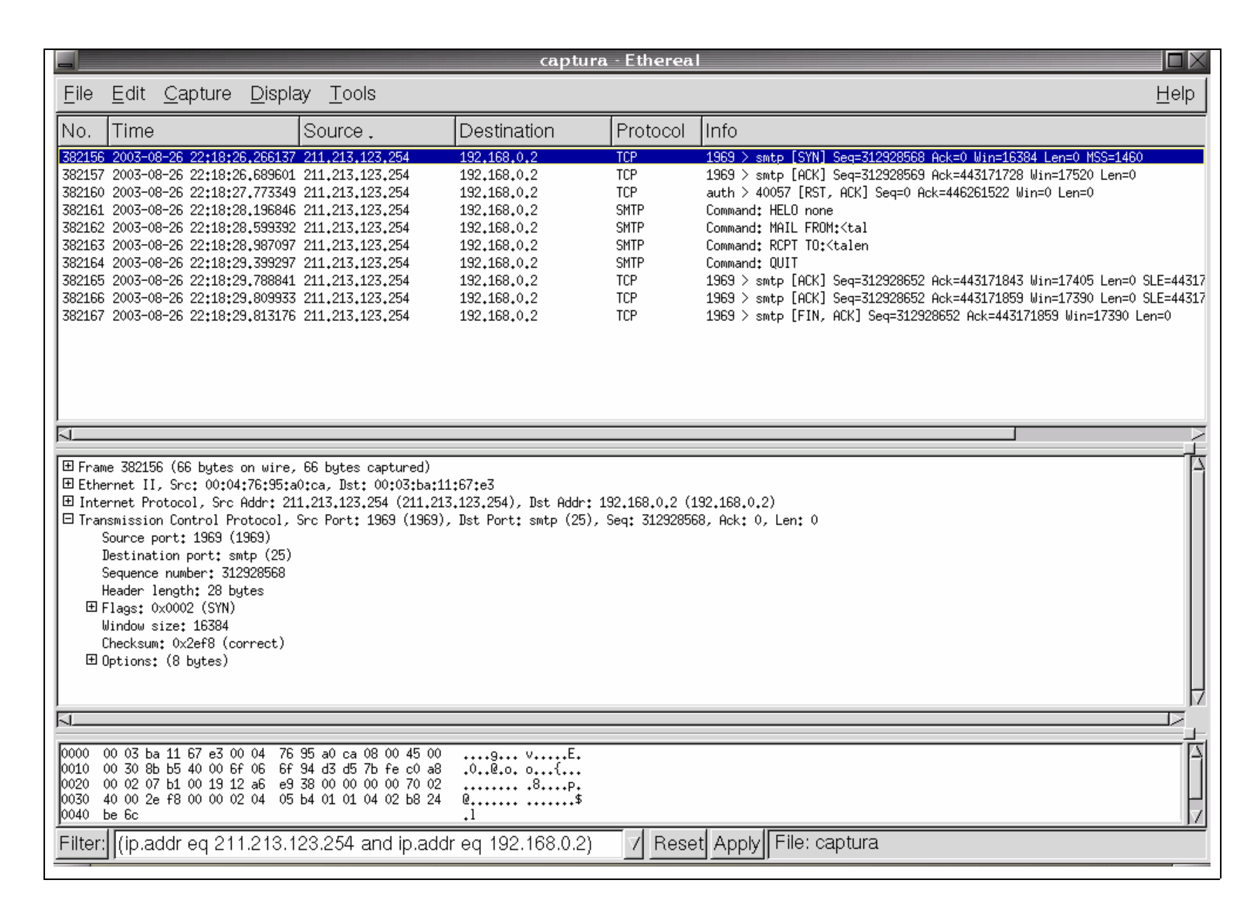


FIG. 5-29: Conexión al servicio MAIL del día 26/08/2003.

En cuanto a las nueve peticiones recibidas al servicio HTTP podemos desglosarlas en dos grupos según su objetivo. En el primer grupo encontramos las peticiones típicas que hacen referencia a la búsqueda de sistemas operativos Windows con servidores IIS mal configurados o no actualizados así como sondeos en búsqueda de capacidades *proxy* de nuestro servidor WWW.

192.168.000.002 217.132.036.071 6 **80** 2824 4150 769 6 6 02:05:07.9312 02:05:13.9552 2 2
192.168.000.002 065.204.029.059 6 **80** 4014 307 503 5 4 04:44:29.1234 04:44:29.7457 2 2
192.168.000.002 204.210.077.240 6 **80** 4395 307 503 5 4 08:14:24.1462 08:14:24.7506 2 2

192.168.000.002 062.255.181.048 6 **80** 18996 307 503 5 4 10:52:30.4965 10:52:32.0998 2 2
192.168.000.002 081.096.155.177 6 **80** 4336 481 519 6 4 12:41:45.7628 12:43:02.1234 2 2
192.168.000.002 081.096.155.177 6 **80** 4545 481 519 6 4 12:49:03.4079 12:50:22.3072 2 2

217.132.36.71 -- [26/Aug/2003:02:05:11 +0200] "GET /default.ida?XXX%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0" 404 270
65.204.29.59 -- [26/Aug/2003:04:44:29 +0200] "GET /scripts/nsiislog.dll" 404 –
204.210.77.240 -- [26/Aug/2003:08:14:24 +0200] "GET /scripts/nsiislog.dll" 404 –
62.255.181.48 -- [26/Aug/2003:10:52:31 +0200] "GET /scripts/nsiislog.dll" 404 –

81.96.155.177 -- [26/Aug/2003:12:41:45 +0200] "OPTIONS / HTTP/1.1" 200 -

En el segundo grupo podemos encontrar unas peticiones con direcciones de origen y destino que no se corresponden a nuestra red local. Incomprensiblemente estas peticiones han llegado a nosotros.

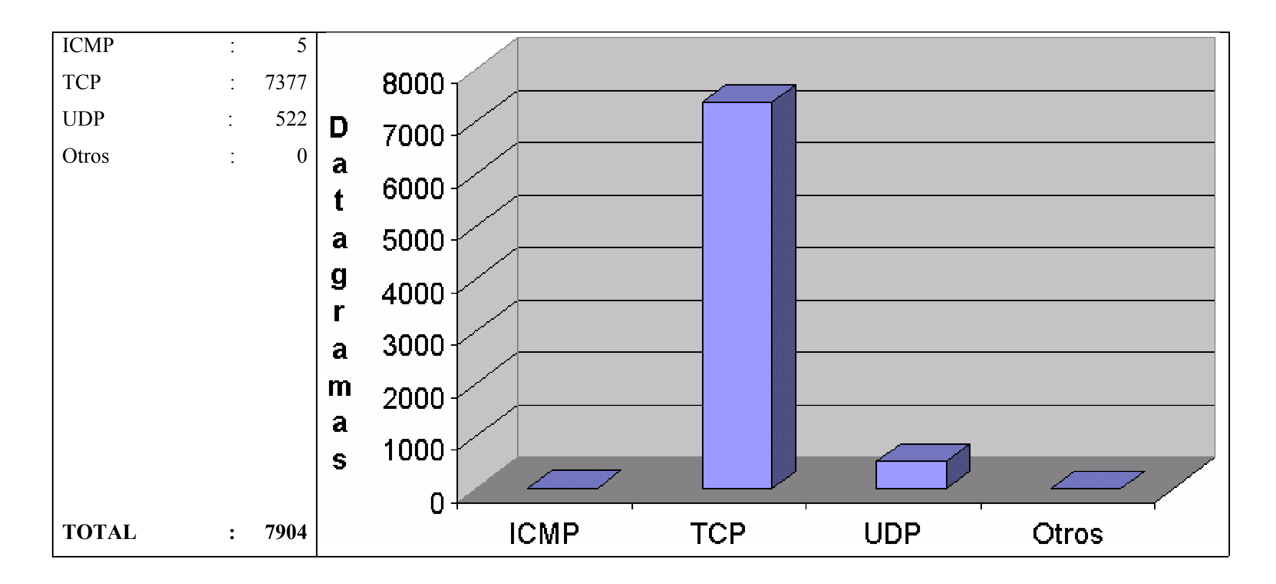
064.028.067.150 216.027.178.155 6 **80** 32771 18396 428911 210 315 19:09:30.1146 19:10:09.8144 2 2
064.028.067.114 216.027.178.155 6 **80** 32772 5929 3906 35 28 19:09:34.8245 19:10:10.4744 2 1
064.028.067.057 216.027.178.155 6 **80** 32773 5460 22176 28 28 19:09:34.8250 19:10:10.3946 2 2

Después de verificar la integridad del servidor tres veces así como comprobar toda la arquitectura desplegada, no hemos podido dar una respuesta satisfactoria a este fenómeno. Podríamos especular asumiendo que de alguna forma desconocida un atacante ha logrado utilizar nuestro servidor WWW como sistema *proxy* y ha lanzado algunas peticiones hacia otras máquinas conectadas a Internet. Sin embargo este punto no es demostrable ya que los logs del servidor WWW no reflejan este comportamiento.

El tráfico recibido por nuestra red el día 26 según el tipo de protocolo (TCP, ICMP, UDP y Otros) al que hacen referencia se puede observar en la figura 5-30.

Como en los días precedentes registramos algunas peticiones ICMP extrañas desde varias direcciones distintas con los puertos de origen 769, 771 o 781 y con destino el puerto local 0.

192.168.000.002 063.147.015.158 **1** 0 781 210 0 3 0 01:00:08.3430 01:00:29.5400 2 2
192.168.000.002 212.033.064.003 **1** 0 771 124 0 1 0 03:30:52.5577 03:30:52.5577 2 2
192.168.000.002 211.128.153.100 **1** 0 771 299 0 1 0 18:36:30.7620 18:36:30.7620 2 2
192.168.000.002 064.014.070.082 **1** 0 769 70 0 1 0 19:00:07.2113 19:00:07.2113 2 2
192.168.000.002 064.014.070.082 **1** 0 769 70 0 1 0 19:30:29.2648 19:30:29.2648 2 2

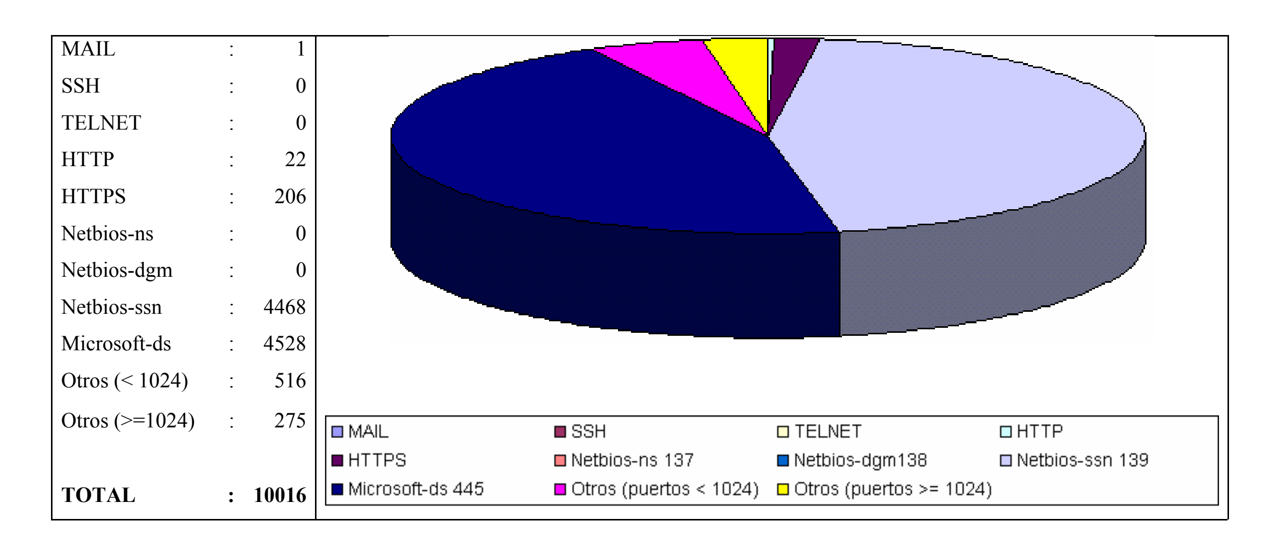


ICMP	:	5
TCP	:	7377
UDP	:	522
Otros	:	0
TOTAL	**:**	**7904**

FIG. 5-30: distribución del tráfico del día 26/08/2003 por protocolo.

5.4.7 Día 27 de Agosto

El último día del estudio (27 de agosto de 2003) se registraron un total de 10.016 paquetes enviados a nuestra red local. El desglose básico del tráfico clasificado por los principales servicios a los que hace referencia podemos observarlo en la figura 5-31.



MAIL	:	1
SSH	:	0
TELNET	:	0
HTTP	:	22
HTTPS	:	206
Netbios-ns	:	0
Netbios-dgm	:	0
Netbios-ssn	:	4468
Microsoft-ds	:	4528
Otros (< 1024)	:	516
Otros (>=1024)	:	275
TOTAL	**:**	**10016**

FIG. 5-31: Clasificación del tráfico del día 27/08/2003 por servicio.

Como a lo largo de la semana, el tráfico de los servicios *Netbios/Microsoft-ds* es el más abundante de los registrados. Se ha recibido una petición de conexión al servicio de MAIL al igual que el día 26.

192.168.000.002 218.187.145.024 6 **25** 4681 524 624 8 9 23:28:42.0097 23:28:48.3334 2 1

En análisis de los logs del sistema (ver figura 5-32) nos revela otro intento de búsqueda de un servidor SMTP mal configurado para enviar correo basura.

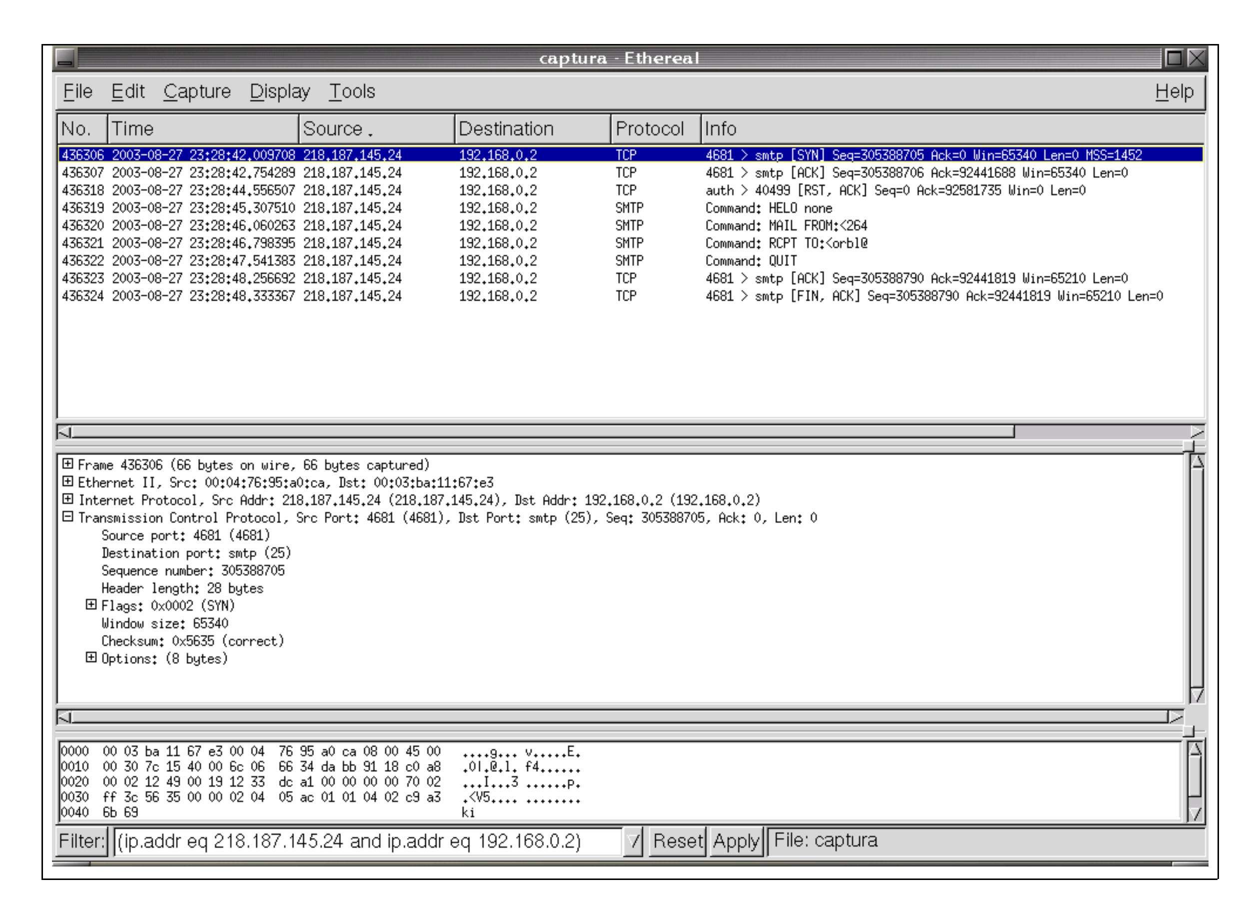


FIG. 5-32: Conexión al servicio MAIL del día 27/08/2003.

En cuanto a las veintidós peticiones recibidas al servicio HTTP podemos desglosarlas en dos grupos según su objetivo. En el primer grupo encontramos las peticiones típicas que hacen referencia a la búsqueda de sistemas operativos Windows con servidores IIS mal configurados o no actualizados y la búsqueda de capacidades *proxy* en nuestro servidor WWW.

192.168.000.002 193.109.252.040 6 **80** 14195 307 503 5 4 10:21:56.9267 10:21:57.6223 2 2
192.168.000.002 217.219.176.067 6 **80** 2455 4150 1260 6 7 15:46:50.3296 15:46:59.3141 2 2
192.168.000.002 066.061.120.188 6 **80** 2504 481 519 6 4 21:00:29.8647 21:01:06.1925 2 2
192.168.000.002 066.061.120.188 6 **80** 2673 481 519 6 4 21:01:32.7518 21:01:59.8359 2 2
192.168.000.002 066.061.120.188 6 **80** 2777 481 519 6 4 21:02:38.3551 21:03:18.9421 2 2
192.168.000.002 066.061.120.188 6 **80** 2837 481 519 6 4 21:03:40.2760 21:04:22.1193 2 2
192.168.000.002 066.061.120.188 6 **80** 2926 481 519 6 4 21:04:43.8393 21:05:04.2428 2 2
192.168.000.002 066.061.120.188 6 **80** 3139 481 519 6 4 21:05:48.5335 21:06:07.8909 2 2
192.168.000.002 066.061.120.188 6 **80** 3362 481 519 6 4 21:06:55.7435 21:07:15.3924 2 2
192.168.000.002 066.061.120.188 6 **80** 3494 481 519 6 4 21:07:59.0118 21:08:22.0239 2 2
192.168.000.002 066.061.120.188 6 **80** 3613 481 519 6 4 21:09:04.2583 21:09:25.8365 2 2
192.168.000.002 066.061.120.188 6 **80** 3804 481 519 6 4 21:10:09.8095 21:10:29.9816 2 2
192.168.000.002 066.061.120.188 6 **80** 4016 481 519 6 4 21:11:17.3926 21:11:53.2347 2 2
192.168.000.002 066.061.120.188 6 **80** 4203 481 519 6 4 21:12:21.0080 21:12:59.1493 2 2
192.168.000.002 066.061.120.188 6 **80** 4317 481 519 6 4 21:13:27.6786 21:13:55.3460 2 2
192.168.000.002 066.061.120.188 6 **80** 4424 481 519 6 4 21:14:30.2513 21:14:48.3605 2 2

193.109.252.40 -- [27/Aug/2003:10:21:57 +0200] "GET /scripts/nsiislog.dll" 404 –

217.219.176.67 -- [27/Aug/2003:15:46:55 +0200] "GET /default.ida?XXX%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0" 404 270

66.61.120.188 - - [27/Aug/2003:21:00:30 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:01:32 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:02:38 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:03:40 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:04:44 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:05:48 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:06:55 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:07:59 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:09:04 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:10:10 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:11:17 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:12:21 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:13:27 +0200] "OPTIONS / HTTP/1.1" 200 -
66.61.120.188 - - [27/Aug/2003:21:14:30 +0200] "OPTIONS / HTTP/1.1" 200 -

En el segundo grupo podemos encontrar una serie de peticiones encargadas de comprobar la existencia de soporte PHP[ℜ] en nuestro servidor WWW. De esta manera suponemos que el atacante buscaba los ficheros típicos de ejemplo (*test.php* o *index.php*) con el objetivo de utilizar algún fallo de seguridad en el PHP para conseguir en control del servidor.

192.168.000.002 217.005.080.066 6 **80** 3878 296 531 5 5 15:16:50.6699 15:16:51.6214 2 2
192.168.000.002 217.005.080.066 6 **80** 3968 358 713 5 5 15:16:55.2389 15:16:55.9981 2 2
192.168.000.002 217.005.080.066 6 **80** 3969 116 124 2 2 15:16:55.2420 15:16:59.3987 2 2
192.168.000.002 217.005.080.066 6 **80** 3970 360 715 5 5 15:16:55.2469 15:16:56.0469 2 2
192.168.000.002 217.005.080.066 6 **80** 3971 357 712 5 5 15:16:55.2551 15:16:56.0941 2 1
192.168.000.002 217.005.080.066 6 **80** 3972 359 714 5 5 15:16:55.2599 15:16:56.1463 2 2

[ℜ] Para más información consultar la bibliografía [WW196].

217.5.80.66 - - [27/Aug/2003:15:16:51 +0200] "GET / HTTP/1.0" 200 0
217.5.80.66 - - [27/Aug/2003:15:16:55 +0200] "GET /index.php HTTP/1.0" 404 268
217.5.80.66 - - [27/Aug/2003:15:16:55 +0200] "GET /phpinfo.php HTTP/1.0" 404 270
217.5.80.66 - - [27/Aug/2003:15:16:55 +0200] "GET /test.php HTTP/1.0" 404 267
217.5.80.66 - - [27/Aug/2003:15:16:55 +0200] "GET /index.php3 HTTP/1.0" 404 269

A continuación podemos observar el desglose del tráfico del día 27 según el tipo de protocolo (TCP, ICMP, UDP y Otros) al que hacen referencia en la figura 5-33.

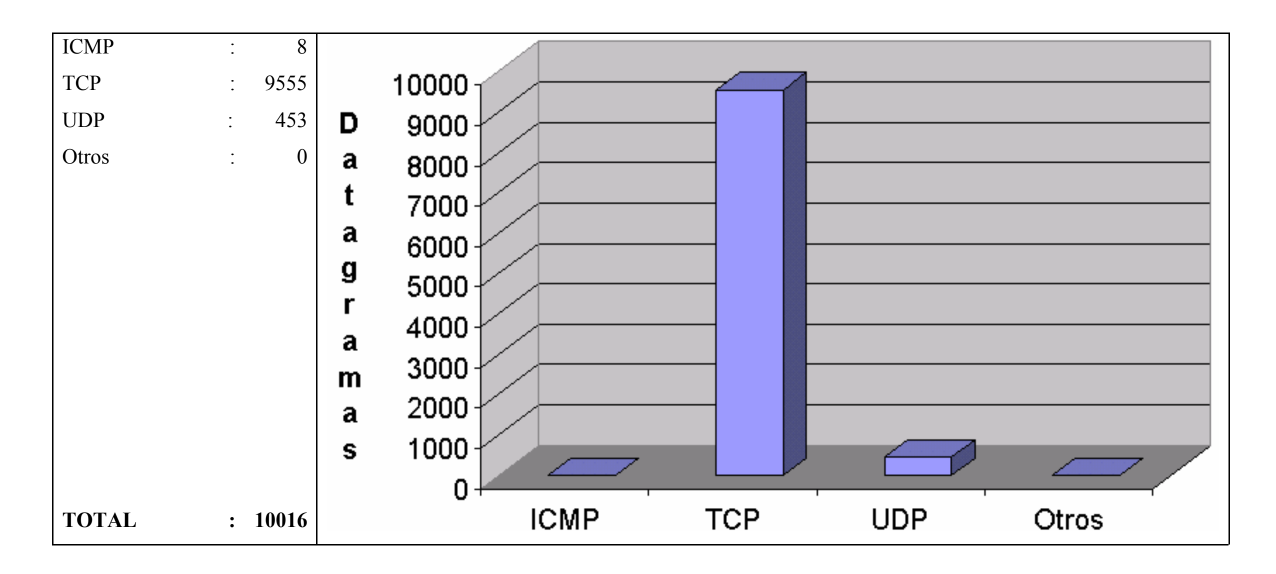


FIG. 5-33: distribución del tráfico del día 27/08/2003 por protocolo.

Como ha ido sucediendo durante toda la semana, volvemos a registrar algunas peticiones ICMP extrañas desde varias direcciones distintas con el puerto de origen 771 o 781 y con destino el puerto 0.

192.168.000.002 212.033.064.003 **1** 0 771 124 0 1 0 00:00:43.0555 00:00:43.0555 2 2
192.168.000.002 207.217.120.013 **1** 0 771 126 0 1 0 01:00:45.2312 01:00:45.2312 2 2
192.168.000.002 212.074.093.014 **1** 0 781 140 0 2 0 02:30:04.7845 02:30:06.7920 2 2
192.168.000.002 212.033.064.003 **1** 0 771 124 0 1 0 10:31:24.2288 10:31:24.2288 2 2
192.168.000.002 063.152.126.150 **1** 0 781 210 0 3 0 17:01:36.9837 17:02:19.1822 2 2
192.168.000.002 198.067.128.010 **1** 0 771 70 0 1 0 18:00:34.0981 18:00:34.0981 2 2
192.168.000.002 209.042.047.067 **1** 0 771 70 0 1 0 18:00:55.8393 18:00:55.8393 2 2
192.168.000.002 200.088.008.230 **1** 0 771 70 0 1 0 22:27:37.3687 22:27:37.3687 2 2

5.4.8 Resumen semanal

Una vez realizado el estudio pormenorizado del tráfico recibido cada día de la semana, pasaremos a presentar el informe resumido de toda la semana dónde destacaremos los elementos más importantes.

En la figura 5-34 podemos apreciar la evolución de la distribución del tráfico que se ha generado durante toda la semana. Podemos observar cómo desde el primer día del experimento el tráfico se ha reducido hasta estabilizarse en unas 11.000 peticiones diarias de media.

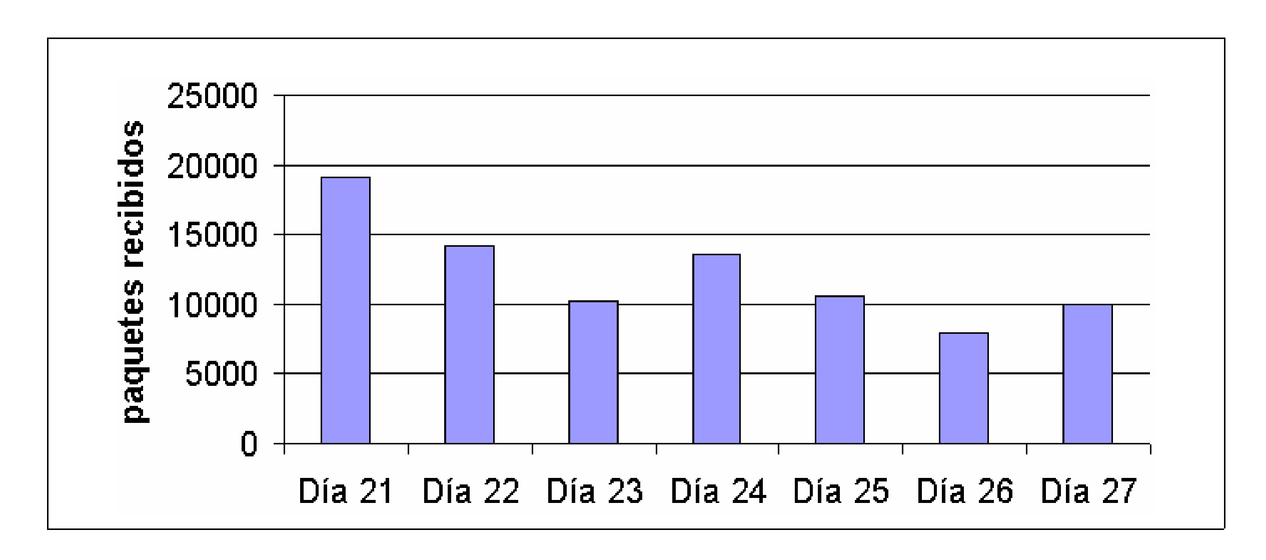
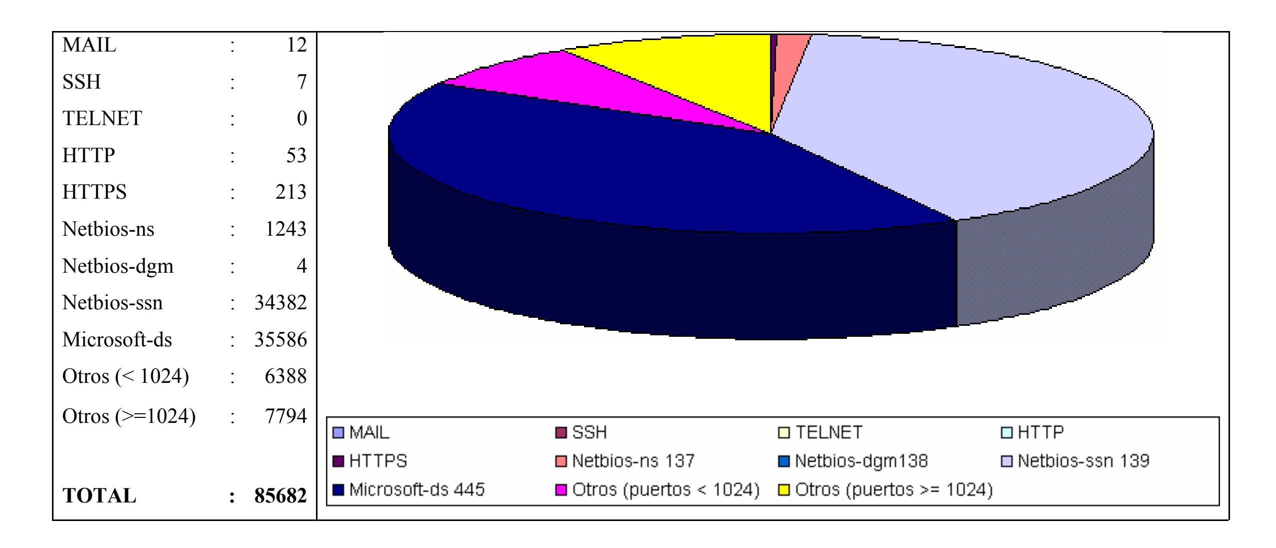


FIG. 5-34: Evolución del tráfico semanal.

El número de peticiones recibidas el día 21 es muy superior (casi el doble) que la media de peticiones semanal. Podemos explicar este hecho basándonos en que prácticamente todas las conexiones existentes en nuestra red de pruebas fueron interrumpidas bruscamente el día 21 para iniciar este experimento. Consecuentemente, los otros extremos de la comunicación enviaban peticiones de servicio durante un tiempo hasta que abandonaban.

En la figura 5-35 podemos observar la distribución semanal de todo el tráfico registrado (85.682 paquetes) por servicio al que hacen referencia.

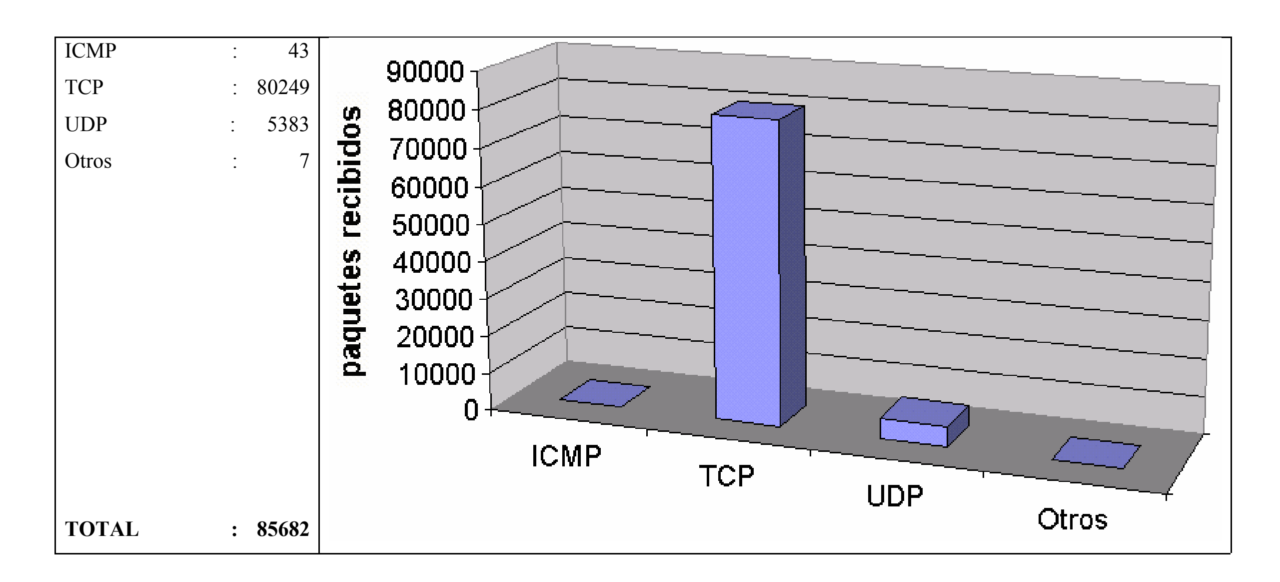


MAIL	:	12
SSH	:	7
TELNET	:	0
HTTP	:	53
HTTPS	:	213
Netbios-ns	:	1243
Netbios-dgm	:	4
Netbios-ssn	:	34382
Microsoft-ds	:	35586
Otros (< 1024)	:	6388
Otros (>=1024)	:	7794
TOTAL	**:**	**85682**

FIG. 5-35: Clasificación semanal del tráfico por servicio.

Podemos observar como las peticiones a los servicios *Netbios/Microsoft-ds* acaparan más del 83% del tráfico registrado. El resto del tráfico se distribuye entre peticiones a puertos distintos de los servicios típicos existentes en el sistema (16%) y minoritariamente (1%) en peticiones y ataques a los servicios de red.

En cuanto a la distribución del tráfico por protocolo (ver figura 5-36), observamos que la mayoría es TCP (93%) y UDP (6%) que mayoritariamente referencia peticiones *Netbios/Microsoft-ds*. El resto corresponde a peticiones anómalas (como las de ICMP).



ICMP	:	43
TCP	:	80249
UDP	:	5383
Otros	:	7
TOTAL	**:**	**85682**

FIG. 5-36: distribución del tráfico semanal por protocolo.

Finalmente en las figuras 5-37 y 5-38 visualizamos una partición del tráfico que ha circulado por nuestra red local (en bytes) según el protocolo al que hace referencia (TCP, UDP, ICMP y otros) y a la dirección de la transmisión que llevaba (si era de entrada hacia nuestra red local o era de salida hacia Internet).

Así mismo podemos observar que la cantidad total de tráfico que ha circulado durante la semana por nuestra red local es de casi 90Mbytes (92.754.552 bytes), lo que realmente es una cantidad de información a tener en cuenta.

TCP out	UDP in	UDP out	ICMP in	ICMP out	Otros in	Otros out	TOTAL
6784084	351798	62059	1025	0	168	0	15064488
7727861	243888	68292	1390	0	0	0	17178892
5053918	69870	50406	947	0	0	0	11312049
7229306	100823	68834	1379	0	0	0	15787368
5259042	138265	62059	264	0	0	0	13671205
4179911	63446	68282	773	0	0	0	8846844
4929469	58604	67750	934	0	0	0	10893706
							92.754.552

FIG. 5-37: Cantidad de bytes que han circulado en la LAN por protocolo.

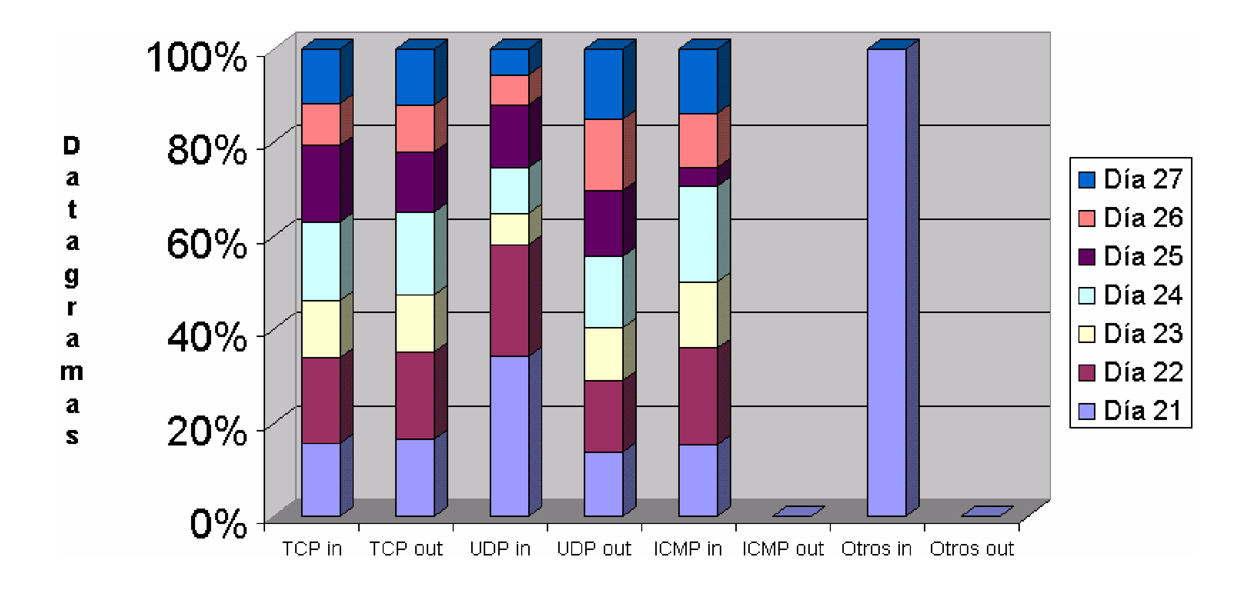


FIG. 5-38: distribución porcentual del tráfico semanal por protocolo.

Del análisis de figura 5-38 podemos deducir rápidamente que generalmente durante toda la semana el tráfico TCP de entrada registrado es prácticamente el mismo que el de salida hacía Internet. Así mismo, el tráfico UDP de entrada suele ser más voluminoso que el de salida, con la excepción de los días 23, 26 y 27 dónde se registra un tráfico de salida muy superior al de entrada.

Prácticamente todo este tráfico (TCP y UDP) registrado hace referencia a interacciones del exterior con los puertos *Netbios/Microsoft-ds* (SAMBA). Coincidiendo con las fechas en las que se realizó este experimento, el virus **W32/BLASTER** tuvo los días de máxima expansión por Internet [WWW197][WWW198][WWW199].

Este virus se caracteriza por aprovechar un fallo (*bug*) del software que controla los servicios de red de Windows, se conecta al puerto TCP/135 (servicio RPC) de los sistemas Microsoft y aprovecha esta vulnerabilidad. Nuestro sistema registró un total de 3086 peticiones al puerto 135.

Finalmente vemos como todo el tráfico ICMP y el clasificado como "Otros" que se ha registrado durante la semana ha sido de volumen similar, con la excepción del día 25 de Agosto dónde el número de peticiones fue netamente inferior a la media. Otro rasgo a comentar es que este tráfico ha sido únicamente de entrada, cosa normal puesto que el análisis de los datagramas nos ha mostrado su naturaleza anómala y por lo tanto carente de respuesta por parte del sistema.

5.5 Conclusiones

Una vez finalizado el experimento y realizado el análisis del tráfico obtenido durante la semana, podemos observar que pese a ser una cantidad a tener en cuenta (casi 90Mbytes) este no representa ni el 5‰ del total teórico semanal. Por otro lado, también es cierto que con una simple petición que tenga éxito por parte de un atacante nuestro sistema puede verse comprometido o utilizado como lanzadera contra otros ordenadores.

En cuanto al análisis del tráfico registrado durante esta semana, cabe destacar la impresionante cantidad de peticiones de los protocolos *Netbios/Microsoft-ds* recibidas, de hecho prácticamente todo el tráfico no deseado registrado pertenecía a esta clase (casi un 83%).

Los protocolos de compartición de recursos de red que utilizan los productos basados en Windows responden de forma automática (sin verificar o autenticar el origen) a cualquier petición realizada. Esta característica lleva a que *hackers* o *blackhats* envíen miles de peticiones indiscriminadas por Internet con el objetivo de conseguir alguna respuesta positiva.

La premisa básica es que la mayoría de sistemas funciona con productos Microsoft, de esta forma la mayoría de peticiones llegará a sistemas Windows (siempre y cuando no exista un firewall o sistema de seguridad de red bien instalado). Por otro lado tenemos que la existencia de fallos (bugs) en el software existente es bastante amplia, lo que permite la existencia de múltiples programas que intentan aprovecharse de sistemas anticuados o no actualizados (como ejemplo tenemos las peticiones en busca de sistemas IIS).

En cuanto al riesgo de ataques directos a nuestra red, hemos podido observar cómo efectivamente si que existen. Los ataques basados en fallos existentes en la implementación de los distintos servicios (SSH, HTTP) han podido ser observados varias veces en nuestro sistema durante la semana.

También hemos recibidos exámenes externos sobre la configuración de nuestros servicios de red (MAIL, HTTP) con el objetivo de utilizarlos como puente de acceso hacia otros ordenadores, ya sea como un *proxy* que permita un acceso anónimo al atacante, un virus o gusano (como en el caso de CodeRed II) o como un foco de propagación de mail no deseado (*spam*).

También hemos recibidos peticiones anómalas y que no hemos podido entender como los datagramas pertenecientes a protocolos desconocidos (ver informe del día 21) o los sospechosos paquetes ICMP dirigidos al puerto 0. Aunque que ciertamente este tráfico es mínimo, existe, lo que puede llegarlo a convertir en un foco de problemas.

Curiosamente no se ha detectado ningún intento de conexión al servicio TELNET, lo que nos lleva a concluir que las conexiones remotas a ordenadores han sido felizmente substituidas por el protocolo SSH.

Hay dos factores fundamentales que han contribuido enormemente al aumento de la inseguridad en Internet:

1. Las conexiones a Internet se han abaratado espectacularmente (podemos conseguir una conexión las 24 horas del día por menos de 40€/mensuales) lo que permite que la cantidad de ordenadores permanentemente conectados sea cada vez mayor.

2. El ancho de banda disponible también ha ido aumentando permitiendo a posibles atacantes el lanzamiento de procesos que sondeen rangos enteros de direcciones de IP en períodos razonablemente cortos de tiempo.

Si bien la cantidad y el éxito de los ataques registrados en nuestra red no han sido preocupantes, sí que debemos concluir este trabajo recomendando la adopción de medidas de seguridad para cualquier red local. Esta recomendación se hace extensible también a los ordenadores personales conectados directamente a Internet (usualmente en el domicilio).

La adopción de algún sistema básico de filtrado de tráfico o puertos[ℜ] debe ser el umbral mínimo de seguridad exigible para cualquier conexión a Internet. Además debemos recomendar la existencia de una estricta política de actualizaciones de software debido a la gran facilidad que tienen actualmente los atacantes para utilizar herramientas automáticas que busquen ordenadores con software "anticuado".

La complejidad de las medidas de seguridad a adoptar depende directamente del tamaño de nuestra red local. El número de ordenadores y servidores existente, la variedad de servicios que ofrezcamos y la cantidad de usuarios del sistema deben guiar nuestra política de seguridad.

[ℜ] Existen cientos de herramientas gratuitas que pueden realizar esta función, por ejemplo los firewall.

Finalmente recordar que **es necesario que la seguridad siempre esté en manos de profesionales**. No hay mayor peligro que un sistema de seguridad mal configurado, ya que crea una falsa sensación de seguridad que será aprovechada por un atacante antes o después.

5.6 RESUMEN

En este último capítulo hemos presentado la parte experimental o práctica de este trabajo. Se ha optado por la realización de un estudio pormenorizado de una conexión permanente a Internet durante siete días.

Inicialmente hemos presentado y analizado los distintos requerimientos funcionales y de estructura que planteaba la monitorización de la red local escogida. Se ha comentado la arquitectura seleccionada para esta prueba, basada en el modelo más representativo de una conexión a Internet. También hemos explicado los distintos servicios que tendríamos funcionando en nuestra red (MAIL, SSH, HTTP…).

Posteriormente se presentaron las diferentes herramientas de monitorización seleccionadas (Apache, Ethereal, IPaudit, MRTG…) que nos permitirán la realización de un estudio efectivo de todo el tráfico que se genere en nuestro sistema.

Finalmente se han presentado y comentado los informes diarios con los resultados e incidencias obtenidas así como un resumen semanal en el que se engloba todo el tráfico registrado durante la semana. Al final del capítulo se realiza una exposición de las conclusiones obtenidas tras la realización de este experimento.

CONCLUSIONES

Seguridad en redes IP

La realización de este trabajo nos ha permitido la obtención de una mayor comprensión de la seguridad existente en las redes IP. No sólo se ha profundizado en el estudio de los protocolos más importantes que permiten el funcionamiento de Internet (IP, UDP, ICMP y TCP) si no que además se han podido observar globalmente, lo que nos ha permitido examinar sus características, relaciones y roles en el transporte de la Información por Internet.

Son numerosos los ataques analizados que se basan en explotar algunas características de los protocolos de comunicación. Estos ataques buscan o bien el cese de las actividades o servicios que presta el ordenador atacado (ataques de denegación de servicio) o bien conseguir un acceso dentro de la máquina que le permita utilizarla a su gusto y conveniencia.

El uso de herramientas de seguridad clásicas basadas en el filtrado simple de los datagramas que circulan por Internet (firewalls) se ha revelado insuficiente ante los ataques organizados. Cuando el atacante es único y está perfectamente identificado (por la dirección IP usualmente) los sistemas básicos de seguridad pueden resultar un muro de defensa relativamente efectivo.

Sin embargo, cuando el atacante no está identificado de una forma explícita o el número de atacantes es desconocido (ataques de denegación de servicio distribuido), estos sistemas no pueden dar la respuesta adecuada.

La proliferación de herramientas automáticas que permiten de una forma sencilla coordinar ataques de cientos o miles de ordenadores simultáneamente, exige sistemas más sofisticados que sean capaces de seguir y entender el flujo de las comunicaciones existentes. El simple filtrado de paquetes aislados del resto del flujo de la comunicación ha dejado de ser efectivo.

También debemos tener en cuenta que muchas veces no basta únicamente con protegernos de las posibles amenazas que provienen de Internet. En el caso de que un sistema fuera comprometido por el motivo que fuese, el resto de ataques que lanzaría el atacante hacia nuestra red serían ataques internos que pasarían desapercibidos.

Los sistemas de detección de intrusos en redes (NIDS) son mecanismos de seguridad pasiva que se encargan de monitorizar todo el tráfico existente en nuestra red, tanto local como remoto. Este tipo de sistemas "inteligentes" nos permitirán incluso la detección de ataques perpetrados por un usuario legítimo de nuestro sistema.

Los sistemas NIDS unen a la capacidad de filtrado del tráfico las posibilidades que brindan la detección por firmas (patrones específicos de ataques conocidos) y el seguimiento de las comunicaciones desde un nivel de flujo de la comunicación.

Las contrapartidas de estos sistemas son principalmente la posibilidad de generar falsos positivos y falsos negativos que pueden desvirtuar la efectividad del sistema. Por otro lado generan una inmensa cantidad de información al tratar con todo el tráfico existente en la red (benigno y maligno) lo que dificulta su post-proceso e interpretación.

Además, para que un sistema de detección de intrusos sea realmente efectivo debe estar perfectamente parametrizado y adaptado a la red en la cual está instalada, lo que implica la existencia de un personal cualificado que regularmente verifique el buen funcionamiento del sistema.

La mejora tecnológica que se produce constantemente en las telecomunicaciones ha llevado a la proliferación de ataques y comportamientos maliciosos que anteriormente eran imposibles. La comprobación de toda una clase A, B o C de direcciones IP empieza a ser factible actualmente en espacios de tiempo cada vez menores. Lo que antes hubiera tardado meses y años, en la actualidad con los anchos de banda mejorando cada día pasa a ser cuestiones de días o incluso horas.

Las antiguas premisas de esconderse tras el anonimato o la falta de documentación pública sobre los aspectos de seguridad o de arquitectura de nuestra red dejan de ser válidos. Los atacantes realizan barridos "ciegos" por toda Internet con el objetivo de conseguir un sistema vulnerable a sus técnicas y métodos.

Los Honeypots son sistemas pasivos que han sido diseñados para ser atacados y/o comprometidos por los atacantes. El cambio de mentalidad que supone la incitación a que ataquen partes concretas de nuestro sistema se basa en la necesidad de conocer y estudiar de una forma fiable los comportamientos y técnicas de los atacantes.

La idea subyacente en el sistema de Honeypot es la de crear un entorno cerrado y acotado dónde los atacantes puedan ser "espiados" con el doble objetivo de conseguir unos nuevos sistemas de protección más seguros y desviar su atención de los sistemas reales. Como el sistema que están atacando está realmente bajo nuestro control, podemos analizar el comportamiento de los atacantes para adoptar las medidas necesarias que nos permitan aplicar una defensa más eficiente de nuestro sistema de producción real.

Las Honeynets son un tipo específico de Honeypots. Estos sistemas persiguen la simulación de una red de ordenadores compuesta de varios equipos y distintos sistemas operativos con el objetivo de que el atacante crea que está atacando una red real.

En la actualidad existen dos generaciones de Honeynets cuya implementación física requiere cada vez de una mayor cantidad de recursos, lo que pueden hacerla inviable para la mayoría de organizaciones. Las Honeynets virtuales permiten abaratar los costes de la arquitectura mediante la instalación de máquinas virtuales en un mismo ordenador que simularán los diferentes sistemas operativos y responderá a tantas direcciones IP como deseemos.

En cuanto a las ventajas añadidas respecto a los sistemas de seguridad tradicionales tenemos que permiten aislar en un entorno controlado a los posibles atacantes y eliminamos la existencia de falsos positivos. Por otro lado, las necesidades de los recursos disminuyen al no existir la necesidad de filtrar o buscar patrones por todo el tráfico y poder crear entornos virtuales en un mismo ordenador.

El riesgo más destacable que introduce el uso de Honeypots y Honeynets es la posibilidad de que un atacante realmente se haga con el control del sistema y lo utilice para atacar a otros ordenadores conectados a Internet, lo que nos obliga a una supervisión constante de estos sistemas.

Parte experimental

El experimento realizado dentro de este trabajo ha consistido en la realización de un estudio estricto del tráfico registrado en una conexión permanente a Internet durante una semana. Nuestro objetivo era el de comprobar si realmente existe en Internet una inseguridad tan grande como las noticias e informes existentes sugieren.

Para llevar a cabo este experimento se realizó un estudio de los diferentes requisitos (software y hardware) necesarios así como de la arquitectura de red a implementar. Se planificaron los distintos servicios que debería proporcionaría nuestra red (SSH, WWW, MAIL…) y se eligieron las diferentes herramientas de monitorización que nos proporcionarían la información deseada.

Del análisis de los datos recogidos durante la semana del 21 al 28 de Agosto de 2003 podemos destacar como prácticamente todo el tráfico existente (83%) hace referencia a los protocolos de servicios de red de Microsoft Windows.

La gran cuota de mercado que posee Microsoft junto con las vulnerabilidades que presentan los sistemas operativos basados en Windows, lleva a que sean los blancos preferidos de los ataques indiscriminados.

Entre el resto del tráfico registrado, hemos podido observar como se recibían distintos tipos de ataques a los servicios que teníamos disponibles en nuestra red. Los ataques más comunes son aquellos que hacen referencia a los servidores WWW y los servicios de MAIL.

La búsqueda de sistemas no actualizados vulnerables a fallos (*bugs*) conocidos es el principal objetivo de los ataques indiscriminados. También se ha observado un cierto número de comprobaciones de la configuración de los distintos servicios de nuestra red (si nuestro servidor WWW ofrece capacidades de *proxy*, si el servidor de MAIL permite enviar correos desde cualquier dirección…) con el objetivo de aprovechar una configuración deficiente de los servicios para sus ataques.

Coincidiendo con este estudio el virus W32/BLASTER tuvo su máximo apogeo, lo que nos permitió observar como recibíamos unos pocos miles de peticiones de ordenadores infectados.

Finalmente y tras analizar todos los resultados obtenidos, llegamos a la conclusión de que si bien sí es cierto que existen una cantidad de peligros que aconsejan siempre la instalación de algún sistema de seguridad en nuestra red u ordenador personal, el volumen y la sofisticación de los ataques distan mucho de crear un Internet caótico.

Líneas futuras de continuación

La realización de este trabajo abarca una gran cantidad de aspectos y conceptos de seguridad de redes relativamente nuevos que presentan una evolución constante.

Los sistemas de detección y bloqueo de ataques DOS/DDOS siguen sin ser efectivos debido a la inexistencia de standares que hayan sido adoptados por sistemas comerciales. La falta de un acuerdo entre fabricantes y organismos internacionales, en parte gracias a la distinta legislación de cada país evita un avance en este campo.

Sin embargo, los sistemas de detección de intrusos en redes (NIDS) es un campo que evoluciona rápidamente y donde más investigadores en seguridad invierten sus esfuerzos. El estudio de nuevas técnicas de detección de intrusos así como la mejora del entendimiento del flujo de las comunicaciones entre ordenadores abre un nuevo campo de estudio denominado IPS (sistemas de prevención de intrusos).

Los Honeypots y Honeynets son técnicas relativamente recientes que últimamente se han convertido en las estrellas de los sistemas de seguridad. Los avances en Honeynets virtuales que permiten la minimización de los recursos destinados así como el refinamiento de sus arquitecturas (generaciones) son un campo de estudio en permanente evolución. La creación y estudio de las Honeynets distribuidas es actualmente una de las líneas futuras con más interés.

En cuanto a la parte experimental, la realización de un estudio más completo, más complejo y con más recursos asociados sobre el tráfico que circula por Internet sería de gran ayuda para observar la evolución de la seguridad en las redes de ordenadores.

Controlar de forma perpetua el tráfico de distintos ordenadores conectados a diferentes puntos de Internet nos permitiría realizar correlaciones y estadísticas más fiables.

BIBLIOGRAFÍA

"...no soy amigo de dar consejos. A nadie le acuchillan en cabeza ajena, más ahí va uno de barato:

Desconfíen siempre vuestras mercedes de quien es lector de un solo libro."

Capitán Alatriste (Del libro Limpieza de sangre).

[DH77] W. Diffie, M. Hellman, "**Privacy and authentication**", IEEE, 1977.

[And80] James P. Anderson, "**Computer Security Threat Monitoring and Surveillance**", James P. Anderson Co., 1980.

[Mas88] J. L. Massey, "**An introduction to contemporary cryptology**", IEEE, 1988.

[Ste90] W. R. Stevens, "**UNIX network programming**", Prenctice-Hall, 1990.

[Sto90] Cliff Stoll, "**The cuckoo's egg: tracking a spy through the maze of computer espionage**", Pocket Books, 1990.

[RH91] J. Rifà i Coma, LL. Huguet i Rotger, "**Comunicación digital**", Masson S.A, 1991.

[Abo93] B. Aboba, "**The Online user's encyclopedia**", Addisson-Wessley, 1993.

[Rif95] J. Rifà i Coma, "**Seguretat Computacional**", Materials, Servei de publicacions UAB, 1995.

[CZ95] D. Brent Chapman, Elizabeth D. Zwicky, "**Building Internet Firewalls**", O'Reilly, 1995.

[Far96] Dan Farmer, "**Security survey of key Internet hosts & various semirelevant reflections**", [WWW158], 1996.

[TH96] S. Talens Oliag, J. Hernandez Orallo, "**Internet. Redes de computadores y sistemas de información**", Paraninfo, 1996.

[Has97] Michael Hasenstein, "**IP Network Address Translation**", [WWW89], 1997.

[Osb97] Sandra Osborne, "**Windows NT Registry**", New Raiders, 1997.

[Esc98] Terry Escamilla, "**Intrusion detection: Network security beyond the firewall**", John Wiley & Sons, 1998.

[Ric98-1] W. Richard Stevens, "**TCP/IP Illustrated Volume 1: The protocols**", Addison-Wessley, 1998.

[Ric98-2] W. Richard Stevens, "**TCP/IP Illustrated Volume 2: The implementation**", Addison-Wessley, 1998.

[Ric98-3] W. Richard Stevens, "**TCP/IP Illustrated Volume 3: TCP transactions**", Addison-Wessley, 1998.

[Sta98] W. Stallings, "**Sistemas operativos**" (2ª edición), Prenctice-Hall, 1998.

[Gar99] Bryan A. Garner, "**Black's law dictionary**", West group (7Th Edition), 1999.

[Dit99] David Dittrich, "**The DOS project's TRINOO distributed denial of service attack tool**", [WWW9], 1999.

[Dit99-2] David Dittrich, "**The STACHELDRAHT distributed denial of service attack tool**", [WWW34], 1999.

[Dit99-3] David Dittrich, "**The TRIBE FLOOD NETWORK distributed denial of service attack tool**", [WWW33], 1999.

[Nor99] Stephen Northcutt, "**Network Intrusion Detection: An analyst's handbook**", New raiders, 1999.

[Sch99] Winn Schwartau, "**Lying to hackers is okay by me**", [WWW116], 1999.

[Sua99] Jaime Suárez Martínez, "**Criptografía**", [WWW54], 1999.

[Alex00] Bryce Alexander, "**Intrusion detection FAQ: port 137 scan**", [WWW180], 2000.

[BT00] Jason Barlow, Woody Thrower, "**TFN2K – An analysis**", [WWW50], 2000.

[Car00] Curtis A. Carver et al., "**A methodology for using intelligent agents to provide automated intrusion response**", IEEE Workshop IAS'00 [WWW70], 2000.

[CMc00] KC Claffy, Sean McCreary, "**Internet Measurement and data analysis: passive and active measurement**", [WWW63], 2000.

[DDL00] Sven Dietrich, David Dittrich, Neil Long, "**An analysis of the SHAFT distributed denial of service tool**", [WWW51], 2000.

[Eve00] Loras R. Even, "**Honeypot system explained**", [WWW124], 2000.

[Gra00] Robert Graham, "**FAQ: Network Intrusion Detection Systems**", [WWW92], 2000.

[Lee00] Wenke Lee et al., "**A data mining and CIDF based approach for detecting novel and distributed intrusions**", [WWW76], 2000.

[Sch00] Bruce Schneier, "**Secrets and lies: Digital security in a networked world**", John Wiley & Sons, 2000.

[Ruo00] Clinton Ruoho, "**Network based Intrusion Detection Systems**" [WWW75], 2000.

[Sto00] Cliff Stoll, "**The Cuckoo's egg**", Pocket Books, 2000.

[Ver00] Gabriel Verdejo Alvarez, "**El protocolo IPv6 y sus extensiones de seguridad IPSec**", proyecto final de carrera en ingeniería superior en informática (UAB), 2000.

[Wan00] Feiyi Wang, "**Vulnerability analysis, intrusion prevention and detection for link state routing protocols**", [WWW77], 2000.

[Wei00] Neal Weinberg, "**Security Survey**", [WWW165], 2000.

[Bru01] Guy Bruneau, "**The history and evolution of Intrusion Detection**", [WWW112], 2001.

[Car01] Curtis A. Carver et al., "**Limiting uncertainly in intrusion response**", IEEE Workshop IAS'01 [WWW71], 2001.

[Chi01] Adhitya Chittur, "**Model generation for an intrusion detection system using genetics algorithms**", [WWW97], 2001.

[Cho01] Tareque Choudhury, "**Honeypots: The trap is set**", [WWW128], 2001.

[Cla01] Michael Clark, "**Virtual Honeynets**", [WWW150], 2001.

[Ein01] Nathan Einwechter, "**An introduction to Distributed Intrusion Detection Systems**", [WWW102], 2001.

[HL01] Jed Haile, Jason Larsen "**Securing an unpatchable webserver…HogWash!**", [WWW110], 2001.

[Hon01] Varios (Honeynet project) [WWW119], "**Know your enemy**", Addison Wesley, 2001.

[GP01] T. M. Gil, M. Poletto, "**MULTOPS: a data-structure for bandwidth attack detection**", 10th Usenix Security Symposium, 2001.

[Kue01] Kirby Kuehl, "**The Honeynet project: Advancements in Honeypot tools**", Computer security 2001 [WWW133], 2001.

[Mar01] William W. Martin, "**Honeypots and Honeynets: Security through deception**", [WWW125], 2001.

[MC01] Ludovic Mé, Cédric Michel, "**Intrusion detection: A bibliography**",SSIR 2001 [WWW109], 2001.

[Mcm01] John F. McMullen, "**Enhance Intrusion Detection with a Honeypot**", TechRepublic [WWW120], 2001.

[Obr01] Eric O'Brien, "**Netbouncer: A practical client-legitimacy-based DDOS defense via ingress filtering**", [WWW155], 2001.

[Rub01] Aviel D. Rubin, "**White-Hat Security Arsenal: Tackling the Tretas**", Addison-Wessley,2001.

[SR01] Lance Spitzner, Marty Roesch, "**The value of Honeypots**", [WWW131], 2001.

[Tim01] Kevin Timm, "**Strategies to reduce false positives and false negatives in NIDS**", [WWW95], 2001.

[Ver01] Gabriel Verdejo Alvarez, "**DDOS: Ataques de denegación de servicio distribuidos**", trabajo de curso de doctorado en el departamento de CCD de la UAB, 2001.

[Vil02] Antonio Villalón Huerta, "**Sistemas de detección de intrusos**", [WWW100], 2002.

[VM01] John Viega, Gary McGraw, "**Building Secure Software: How to Avoid Security Problems the Right Way**", Addison-Wessley, 2001.

[BP02] Reto Baumann, Christian Plattner, "**Honeypots**", Diploma Thesis at Swiss Federal Institute of technology (Zurich) [WWW118], 2002.

[Dit02] David Dittrich, "**Root kits**", [WWW48], 2002.

[DLD02] Ralph Droms, Ted Lemon, Ralph E. Droms, "**The DHCP handbook**", SAMS 2nd Edition, 2002.

[Fran02] Iain Franklin, "**Intrusion Detection Systems (IDS)**", [WWW111], 2002.

[Gib02] Steve Gibson, “**DRDoS: Distributed Reflection Denial of Service**”, [WWW57], 2002.

[HFC02] B.L Hutchings, R. Franklin, D. Carver, “**Assisting Network Intrusion Detection with Reconfigurable Hardware**”, IEEE FCCM’02 [WWW93], 2002.

[Kid02] Angus Kidman, “**Like hackers to a Honeypot**”, [WWW129], 2002.

[Kue02] Kirby Kuehl, “**Intrusion Deception**”, Networld+Interop’02 [WWW132], 2002.

[KMR02] A. D. Keromytis, V. Misra, and D. Rubenstein, “**SOS: Secure Overlay Services.**”, SIGCOMM, 2002.

[Liu02] Cricket Liu, “**DNS & BIND Cookbook**”, O’Reilly & Associates, 2002.

[Mah02] R. Mahajan et al., “**Controlling high bandwidth aggregates in the network.**”, ACM Computer Communications Review, 2002.

[Pou02] Kevin Poulsen, “**Wi-Fi Honeypots a new hacker trap**”, [WWW142], 2002.

[Pu02] Chris Prosise, Saumil Udayan Shah, “**Hacker’s tricks to avoid detection**”, [WWW], 2002.

[Ran02] Marcus J. Ranum, “**A whirldwind introduction to honeypots**”, [WWW127], 2002.

[Spi02] Lance Spitzner, “**Honeypots: tracking Hackers**”, Addison Wesley professional, 2002.

[ZDD02] Laurie Zirkle, George Drake, Dave Dittrich, “**Incident handling step by strep: UNIX Trojan programs**”, [WWW41], 2002.

[Bor03] Tim Boreham, “**Security survey says life’s breach**”, [WW162], 2003.

[Des03] Neil Desai, “**Intrusion prevention systems: The next step in the evolution of IDS**”, [WWW107], 2003.

[Gre03] Robyn Greenspan, “**Security breaches rage through Asia pacific**”, [WWW115], 2003.

[Hon03] Varios (Honeynet project) [WWW119],”**Know your enemy: Honeynets**”, [WWW119], 2003.

[Hon03-2] Varios (Honeynet project) [WWW119],”**Honeynet: Definitions, Requirements and Standards**”, [WWW146], 2003.

[Hon03-3] Varios (Honeynet project) [WWW119],”**GEN II Honeynets**”, [WWW147], 2003.

[Hon03-4] Varios (Honeynet project) [WWW119],”**Defining virtual Honeynets**”, [WWW151], 2003.

[Lar03] Bob Larson, “**Introduction to Network Security**”, AINAC [WWW152], 2003.

[Lev03] John Levine et al., **“The use of honeynets to detect exploited systems across large enterprise networks**”, IEEE Workshop on Information Assurance [WWW121], 2003.

[Lev03-2] John Levine et al., “**The use of Honeynets to detect exploited systems across large enterprise networks**”, IEEE 2003 Procedings: Workshop of information assurance [WWW135], 2003.

[MP03] Jelena Mirkovic, Peter Reiher,“**A taxonomy of DDOS attacks and defense mechanisms**”, ACM [WWW153], 2003.

[MP03-1] Jelena Mirkovic, Peter Reiher, “**Attacking DDOS at the source**”, [WWW153], 2003.

[MP03-2] Jelena Mirkovic, Peter Reiher, “**Challenges and principles of DDOS defense**”, NCA [WWW153], 2003.

[Pou03] Kevin Poulsen, "**Use a Honeypot, Go to prison?**", [WWW139], 2003.

[Pro03] Niels Provos, "**The Honeynet project: Virtual Honeypots**", [WWW144], 2003.

[Spi03] Lance Spitzner, "**Honeypots: Definitions and value of honeypots**", [WWW122], 2003.

[Spi03-2] Lance Spitzner, "**Honeypots: Simples, cost-effective detection**", [WWW136], 2003.

[Spi03-3] Lance Spitzner, "**Honeytokens: The other Honeypot**", [WWW138], 2003.

[Spi03-4] Lance Spitzner et al., "**Honeypots: Are They Illegal?**", [WWW140], 2003.

[Tan03] Mattew Tanase, "**IP Spoofing: An introduction**", [WWW108], 2003.

[TCE03] Jay Ts, David Collier-Brown, Robert Eckstein, "**Samba pocket reference**", O'reilly & associates, 2003.

BIBLIOGRAFÍA WWW

[RFC] http://www.ietf.org/rfc/
http://www.faqs.org/rfcs/index.html
ftp://ftp.rediris.es/pub/rfc

[WWW1] http://www.diccionarios.com

[WWW2] http://www.cddhcu.gob.mx/camdip/foro/jalisco/tipi_del.htm

[WWW3] http://www.sitios.uach.cl/caminosfor/CristianSalazar/SIA/AI.html

[WWW4] http://spisa.act.uji.es/SPI/TEORIA/Temario/tema1/

[WWW5] http://www.zakon.org/robert/internet/timeline/

[WWW6] http://www.nua.ie/surveys/how_many_online/

[WWW7] http://www.netvalley.com/intvalstat.html

[WWW8] http://www.dcc.uchile.cl/~hsalgado/dos/siframes.htm

[WWW9] http://staff.washington.edu/dittrich/misc/trinoo.analysis

[WWW10] http://www.cert.org/advisories/CA-1999-17.html

[WWW11] http://www.cert.org/advisories/CA-2000-01.html

[WWW12] http://www.protocols.com/pbook/tcpip1.htm

[WWW13] http://www.synapse.de/ban/HTML/P_TCP_IP/Eng/P_tcp_11.html

[WWW14] http://www.cisco.com/en/US/products/hw/vpndevc/ps976/products_user_guide_chapter09186a008007d173.html

[WWW15] http://www.infowar.com/hacker/hack_072397a.html-ssi

[WWW16] http://www.symonds.net/~deep/stuff/tekmail/issue35.php

[WWW17] http://www.iss.net/security_center/advice/Exploits/Ports/

[WWW18] http://isis.poly.edu/projects/CS393/denial.pdf

[WWW19] http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ip.htm

[WWW20] http://www.stanford.edu/class/ee380/Slides/000223/sld001.htm

[WWW21] http://www.cert.org

[WWW22] http://lucas.hispalinux.es/Tutoriales/doc-tutorial-recomendaciones-seguridad/doc-tutorial-recomendaciones-seguridad.html

[WWW23] http://www.net-security.org

[WWW24] http://www.securityfocus.com

[WWW25] http://standards.ieee.org/regauth/oui/index.shtml

[WWW26] http://whatis.techtarget.com/definition/0,,sid9_gci212506,00.html

[WWW27] http://whatis.techtarget.com/definition/0,,sid9_gci213780,00.html

[WWW28] http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci214257,00.html

[WWW29] http://www.redes.unb.br/material/arquiteturaeprotocolosderede/AulaARP-RARP.pdf

[WWW30] http://www.geocities.com/ddragos2002/Frames.htm

[WWW31] http://cisco.netacad.net

[WWW32] http://www.erg.abdn.ac.uk/users/gorry/course/lan-pages/mac-vendor-codes.html

[WWW33] http://staff.washington.edu/dittrich/misc/tfn.analysis

[WWW34] http://staff.washington.edu/dittrich/misc/stacheldraht.analysis

[WWW35] http://packetstorm.securify.com/distributed/TFN2k_Analysis.htm

[WWW36] http://www.cert.org/reports/dsit_workshop.pdf

[WWW38] http://www.ddosworld.com/

[WWW39] http://grc.com/dos/grcdos.htm

[WWW40] http://www.cisco.com/warp/public/707/newsflash.html

[WWW41] http://www.sans.org/y2k/DDoS.htm

[WWW42] http://staff.washington.edu/dittrich/misc/ddos/

[WWW43] http://www.koc.net/koc_kurumsal/destek/signatures70/attacks/Snork_Attack.htm

[WWW44] http://www.anml.iu.edu/ddos/types.html

[WWW45] http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci213591,00.html

[WWW46] http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci557336,00.html

[WWW47] http://www.cert.org/incident_notes/IN-99-04.html

[WWW48] http://staff.washington.edu/dittrich/misc/faqs/rootkits.faq

[WWW49] http://staff.washington.edu/dittrich/misc/tfn.analysis.txt

[WWW50] http://packetstormsecurity.com/distributed/TFN2k_Analysis-1.3.txt

[WWW51] http://www.sans.org/y2k/shaft.htm

[WWW52] http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt

[WWW53] http://www.counterpane.com/blowfish.html

[WWW54] http://webs.ono.com/usr005/jsuarez/cripto.htm

[WWW55] http://www.ddosworld.com/

[WWW56] http://grc.com/dos/grcdos.htm

[WWW57] http://grc.com/dos/drdos.htm

[WWW58] http://www.seifried.org/security/ports/

[WWW59] http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci295031,00.html

[WWW60] http://www.tripwire.org/

[WWW61] http://www.sans.org/resources/idfaq/

[WWW62] http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028abstract.html

[WWW63] http://www.caida.org/outreach/papers/1999/Nae4hansen/Nae4hansen.html

[WWW64] http://www.caida.org/

[WWW65] http://www.caida.org/outreach/papers/

[WWW66] http://www.cert.org/tech_tips/packet_filtering.html

[WWW67] http://seclab.cs.ucdavis.edu/projects/vulnerabilities/doves/

[WWW68] http://www.ids-world.com/newsletters/00_34q/pwr_help.htm

[WWW69] http://www.marimba.com/doc/461/cent_logging/abouta.htm

[WWW70] http://www.itoc.usma.edu/marin/Wshop/Papers2000/TP1_1.pdf

[WWW71] http://www.itoc.usma.edu/Workshop/2001/Authors/Submitted_Abstracts/paperT3A1(56).pdf

[WWW72] http://www.infosecuritymag.com/2002/jun/cover.shtml

[WWW73] http://wwwcs.dongguk.ac.kr/~withlhw/IDS_IEEE/00738563.pdf

[WWW74] http://www1.cs.columbia.edu/ids/concept/

[WWW75] http://itd.colorado.edu/huan5831/docs/Lecture%204%20IDS.pdf

[WWW76] http://citeseer.nj.nec.com/309543.html

[WWW77] http://www.cs.ucdavis.edu/~wu/publications/fwphd.pdf

[WWW78] http://www.checkpoint.com/products/downloads/opsec_whitepaper.pdf

[WWW79] http://www.isi.edu/gost/cidf/

[WWW80] http://www.darpa.mil/ito/

[WWW81] http://www.ietf.org/html.charters/idwg-charter.html

[WWW82] http://www.checkpoint.com/press/1998/cvp111198.html

[WWW83] http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028app-e.html

[WWW84] http://www.checkpoint.com/press/1998/ansa1098.html

[WWW85] http://ansa.iss.net

[WWW86] http://www1.cs.columbia.edu/ids/HAUNT/readme.html

[WWW87] http://www.itsecurity.com/dictionary/dictionary.htm

[WWW88] http://www.nipc.gov/about/pdd63.htm

[WWW89] http://www.suse.de/~mha/linux-ip-nat/diplom/nat.html

[WWW90] http://www.homenethelp.com/web/explain/about-NAT.asp

[WWW91] http://www.squid-cache.org/

[WWW92] http://www.robertgraham.com/pubs/network-intrusion-detection.html

[WWW93] http://www.computer.org/proceedings/fccm/1801/18010111abs.htm

[WWW94] http://info.acm.org/crossroads/xrds2-4/intrus.html

[WWW95] http://www.securityfocus.com/infocus/1463

[WWW96] http://www.securityfocus.com/infocus/1477

[WWW97] http://www1.cs.columbia.edu/ids/publications/gaids-thesis01.pdf

[WWW98] http://www.silicondefense.com/software/spice/index.htm

[WWW99] http://csiannual.com/pdf/i8.pdf

[WWW100] http://andercheran.aiind.upv.es/toni/personal/transpas-ids.pdf

[WWW101] http://link.springer.de/link/service/journals/10207/bibs/1001001/10010014.htm

[WWW102] http://www.securityfocus.com/infocus/1532

[WWW103] http://www.takedown.com

[WWW104] http://www.kevinmitnick.com/home.html

[WWW105] http://ccia.ei.uvigo.es/docencia/SSI/DeteccionDeIntrusos.pdf

[WWW106] http://www.gulker.com/ra/hack/tsattack.html

[WWW107] http://www.securityfocus.com/infocus/1670

[WWW108] http://www.securityfocus.com/infocus/1674

[WWW109] http://www.raid-symposium.org

[WWW110] http://www.securityfocus.com/infocus/1208

[WWW111] http://www.itsecurity.com/asktecs/jul3302.htm

[WWW112] http://www.sans.org/rr/intrusion/evolution.php

[WWW113] http://www.uniforum.chi.il.us/slides/ddos/A%20Brief%20History%20of%20Distributed%20Denial%20of%20Service.PPT

[WWW114] http://www.robertgraham.com/op-ed/magic-ddos.html

[WWW115] http://cyberatlas.internet.com/big_picture/applications/article/0,,1301_2206701,00.html#table

[WWW116] http://www.nwfusion.com/newsletters/sec/0607sec2.html

[WWW117] http://www.epp.cmu.edu/19-601/

[WWW118] http://security.rbaumann.net/download/diplomathesis.pdf

[WWW119] http://project.honeynet.org/papers/

[WWW120] http://www.techrepublic.com/article_guest.jhtml?id=r00220010412mul01.htm&fromtm=e036

[WWW121] http://www.tracking-hackers.com/papers/gatech-honeynet.pdf

[WWW122] http://www.spitzner.net/honeypots.html

[WWW123] http://www.honeynet.org/papers/stats/

[WWW124] http://www.sans.org/resources/idfaq/honeypot3.php

[WWW125] http://www.sans.org/rr/papers/4/41.pdf

[WWW126] http://www.honeypots.net/

[WWW127] http://www.certconf.org/presentations/2002/Tracks2002Expert_files/HE-1&2.pdf

[WWW128] http://www.cyberguard.com/events/seminar_recaps/bracknell/tareque_choudhury.pdf

[WWW129] http://www.smh.com.au/articles/2002/08/10/1028158034383.html

[WWW130] http://www.tracking-hackers.com/misc/faq.html

[WWW131] http://www.securityfocus.com/infocus/1492

[WWW132] http://winfingerprint.sourceforge.net/presentations/

[WWW133] http://www.seguridad2001.unam.mx/security2001/seg2001i.php

[WWW134] http://www.all.net/dtk/dtk.html

[WWW135] http://www.tracking-hackers.com/papers/gatech-honeynet.pdf

[WWW136] http://www.securityfocus.com/infocus/1690

[WWW137] http://www.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/docs/as400v35/docs/topdmz.html

[WWW138] http://www.securityfocus.com/infocus/1713

[WWW139] http://www.securityfocus.com/news/4004

[WWW140] http://www.securityfocus.com/infocus/1703

[WWW141] http://standards.ieee.org/wireless/

[WWW142] http://www.securityfocus.com/news/552

[WWW143] http://www.incident-response.org/WISE.htm

[WWW144] http://niels.xtdnet.nl/papers/honeynet/

[WWW145] http://www.webopedia.com/TERM/H/honeynet.html

[WWW146] http://project.honeynet.org/alliance/requirements.html

[WWW147] http://project.honeynet.org/papers/gen2/

[WWW148] http://project.honeynet.org/papers/honeynet/tools/

[WWW149] http://www.tracking-hackers.com/conf/honeypots-1.1.ppt.zip

[WWW150] http://www.securityfocus.com/infocus/1506

[WWW151] http://project.honeynet.org/papers/virtual/

[WWW152] http://ainac.tgm.ac.at/download/Intro_to_Network_Security_BLarson.ppt

[WWW153] http://www.cs.ucla.edu/~sunshine/publications/

[WWW154] http://www.cs3-inc.com/mananet.html

[WWW155] http://www.networkassociates.com/us/_tier0/nailabs/_media/research_projects/development_solutions/netbouncer_presentation.pdf

[WWW156] http://www.dhcp.org/

[WWW157] http://us1.samba.org/samba/samba.html

[WWW158] http://www.trouble.org/survey/

[WWW159] http://www.pwcglobal.com/extweb/service.nsf/docid/F017BDD30988361085256A84004EBDB8

[WWW160] http://www.mcafee.com

[WWW161] http://www.symantec.com/

[WWW162] http://australianit.news.com.au/articles/0,7204,6653406%5e15317%5e%5enbv%5e15306,00.html

[WWW163] http://www.cert.org/stats/cert_stats.htm

[WWW164] http://www.census.gov/eos/www/css/css.html

[WWW165] http://www.nwfusion.com/research/2000/0508secsurvey.html

[WWW166] http://www.apache.org

[WWW167] http://news.netcraft.com/archives/web_server_survey.html

[WWW168] http://www.ethereal.com/

[WWW169] http://ipaudit.sourceforge.net/

[WWW170] http://ipaudit.sourceforge.net/ipaudit-web/

[WWW171] http://people.ee.ethz.ch/~oetiker/webtools/mrtg/

[WWW172] http://www.insecure.org/nmap

[WWW173] http://www.tcpdump.org

[WWW174] http://sourceforge.net/projects/tcpdump/

[WWW175] http://tcpreplay.sourceforge.net/

[WWW176] http://www.iana.org/assignments/port-numbers

[WWW177] http://www.sans.org/resources/tcpip.pdf

[WWW178] http://www.hispasec.com/unaaldia/1768

[WWW179] http://www.esecurityplanet.com/alerts/article.php/2107481

[WWW180] http://www.sans.org/resources/idfaq/port_137.php

[WWW181] http://www.broadbandreports.com/faq/177

[WWW182] http://www.upseros.net/portscan/portscan.php

[WWW183] http://www.google.com/press/zeitgeist/zeitgeist-jun03.html

[WWW184] http://www.leprechaun.com.au/VirusBuster/Alert.asp?NewsID=141

[WWW185] http://monaco.cis.temple.edu/~gyan/Report_5.htm

[WWW186] http://www.webmasterworld.com/forum39/1274.htm

[WWW187] http://lists.virus.org/incidents-0201/msg00130.html

[WWW188] http://login.caida.org/pipermail/cflowd/2003-February/000296.html

[WWW189] http://bugs.php.net/bug.php?id=19113

[WWW190] http://blogs.bwerp.net/archives/2003/06/26/lock_your_doors/

[WWW191] http://lists.insecure.org/lists/fulldisclosure/2003/Jun/0049.html

[WWW192] http://wave.prohosting.com/aosrk/1337.html

[WWW193] http://www.google.com/preferences?hl=xx-hacker

[WWW194] http://archives.neohapsis.com/archives/incidents/2003-07/0220.html

[WWW195] http://secinf.net/misc/Hackers_Tricks_to_Avoid_Detection_.html

[WWW196] http://www.php.net/

[WWW197] http://www.tik.ee.ethz.ch/~ddosvax/blaster/

[WWW198] http://www.cert.org/advisories/CA-2003-20.html

[WWW199] http://www.hispasec.com/unaaldia/1751



Gabriel Verdejo Alvarez

Barcelona, Septiembre de 2003.